



Domino

The Power to Connect People — Easily, Securely, Reliably

5
RELEASE

Domino Enterprise Integration Guide

Release 5.0

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or part, without the prior written consent of Lotus Development Corporation.

© Copyright 1998 Lotus Development Corporation, an IBM subsidiary
55 Cambridge Parkway
Cambridge, MA 02142

All rights reserved. Printed in the United States of America.

Lotus and Lotus Notes are registered trademarks, and LotusScript, Lotus NotesView, Domino Connectors and Domino Enterprise Connection Services are trademarks of Lotus Development Corporation.

AIX, AS/400, DB2, DPROPR, OS/400, IBM, MVS, and OS/2 are registered trademarks, and DB2/2 and RS/6000 are trademarks of International Business Machines Corporation.

All other trademarks are property of their respective owners.

Contents

Part 1: Domino Connectors Setup Guide

Chapter 1 Introduction 1

Overview	1
Organization of this Manual	2
Related Documentation	3
Domino Enterprise Connection Services..	3
LotusScript Extension for Lotus	
Connectors.....	3
Enterprise Integrator Documentation	3
Other Documentation	4

Chapter 2 LCTEST 5

Overview	5
Supported Data Sources	5
Requirements.....	5
Running LCTEST.....	6

Chapter 3 DB2 Connectivity 9

Requirements for DB2 Connectivity	9
DB2 Connectivity Test.....	10
Server Connectivity to DB2	11
Connectivity to DB2 on AS/400.....	12

Direct DB2/400 Connectivity.....	12
ODBC DB2/400 Connectivity	13
Connection for DB2 CAE and DDCS	14
Getting Started	14
Sample Steps for DB2 Configuration.....	15
When to use IBM Distributed Database	
Connection Services (DDCS).....	17
Sample Steps for using DDCS	18
MDI Gateway.....	19

Chapter 4 EDA/SQL Connectivity 21

Requirements for EDA/SQL Connectivity ..	21
EDA/SQL Connectivity Test.....	22

Chapter 5 ODBC Connectivity..... 23

Requirements for ODBC Connectivity.....	23
ODBC Connectivity Test	24

Chapter 6 Oracle Connectivity 25

Requirements for Oracle Connectivity	25
Oracle Connectivity Test.....	26

Chapter 7 Sybase Connectivity..... 27

Requirements for Sybase Connectivity	27
Sybase Connectivity Test	28

Part 2:

Domino Enterprise Connection Services User's Guide

Chapter 1

Introduction 31

Organization of this Manual	31
Introduction to Domino Enterprise Connection Services	32
Supported Data Sources	33
Contact and Support Information	34
Enterprise Integration Sales Support	34
Telephone Support	34
Contacting 3 rd Party Support	34

Chapter 2

DECS Administrator 35

The DECS Administrator	35
DECS Navigator.....	36
Connections View	37
RealTime Activities View	38
Menu Commands	40

Chapter 3

Defining Lotus Connections 43

Defining Lotus Connections Using the Wizard.....	43
Using the Override Button	46
Example of a Completed Lotus Connection document	47

Chapter 4

Creating RealTime Activities49

Overview.....	49
Creating a RealTime Activity Using the Wizard.....	49
Creating a RealTime Activity without the Wizard.....	49
Usage Requirements.....	50

How to Create a RealTime Activity Using the Wizard

Step 1 – Click on the New RealTime Activity Icon	51
Step 2 – Select the Domino Database to be Monitored	52
Step 3 – Select the Notes Form to be Monitored	53
Step 4 – Select the External Data Source	54
Step 5 - Map Key Field(s) and Data Field(s)	55
Step 6 – Select the Document Events to Monitor	56
Step 7 – Name the RealTime Activity.....	57

How to Create a RealTime Activity

Step 1 – Click on the New RealTime Activity icon.....	59
Step 2 – Enter a Name for the Activity ...	60
Step 3 – Select the Notes Application.....	60
Step 4 – Select the External Data Source	60
Step 5 – Map Key and Data Fields.....	61
Step 6 – Select Event(s) to Monitor	61
Step 7 – Select Options	61
Step 8 – Save and Close the Document...	61
Step 9 – Process the RealTime Activity ..	62

RealTime Activity Options

General Options	64
Document Creation Options.....	68
Document Open Options.....	68
Document Update Options.....	69
Document Deletion Options.....	70

Logging of RealTime Activity Status.....

Chapter 5 Building the Notes Application .73

Building the Notes Application for a RealTime Activity	73
Example Application	74
Example Application Design View	75
Example RealTime Activity	77

Chapter 6 Examples using RealTime Activity Options.....79

Using Filter Formulas.....	79
Saving Data to an External Data System	79
Accessing Different External Sources using the Same Notes Application	80
Using Data Storage Options	80
Building Views.....	80
Storing a Static Copy of External Data...	80
Using Monitor Orders	81
Using Stored Procedures	83
Using the Stored Procedures in SQL	85
Stored Procedure Definitions	85

Chapter 7 RealTime Dynamic Queries89

Overview	89
RealTime Dynamic Queries from Web Clients	90
Overview of Steps	90
RealTime Dynamic Queries from Notes Clients	93

Appendix A Configuration and Troubleshooting 97

NOTES.INI Variables.....	97
Installing NotesPump 2.5a after DECS	98
Using DECS with Notes Clients Older than Release 4.62	98
Using DECS on Solaris Platforms.....	98
Error Messages.....	100
Cannot use field ['FIELDNAME'] as both a key and a data field.....	100
Failure accessing shared RealTime Activities table	100
Failure encountered in monitoring process -- ERROR MESSAGE	100
Unexpected internal failure in RealTime monitoring	100
Update of key field ['FIELDNAME'] is not permitted	100
This record has changed in the backend database since being opened - action cancelled	100
Cannot locate corresponding external record	101
Failure compiling Filter Formula: FORMULA COMPILATION ERROR	101
Failure compiling Pre-Open Formula: FORMULA COMPILATION ERROR	101
Failure compiling Post-Update Formula: FORMULA COMPILATION ERROR	101
Failure compiling Post-Create Formula: FORMULA COMPILATION ERROR	101
Failure compiling Post-Delete Formula: FORMULA COMPILATION ERROR	101
"Unknown OS error: libdecsext.*"	102
"DECS Server addin task initialization failed"	102
"DECS Server is unable to allocate addin task resources"	102

"DECS Server cannot connect to external system"	102
"DECS Server cannot find external table/metadata"	102
"DECS Server cannot find external procedure/transaction"	102
"DECS Server error retrieving external record"	102
"DECS Server error inserting external record"	102
"DECS Server error updating external record"	103
"DECS Server error deleting external record"	103
"DECS Server cannot locate the corresponding record in the external system"	103
"DECS Server unable to update document due to key field changes; changes to key fields have been disabled"	103
"DECS Server unable to update document due to conflict; the external record has been modified since being opened"	103
"DECS Server data overflow accessing external record"	103

PART 3:

LotusScript Extension for Domino Connectors Reference Guide

Chapter 1

Introduction107

Organization of this manual.....	107
Overview of the LotusScript Extension for Domino Connectors.....	109
Connectivity Software Requirements ...	110
Registration and Loading of the	
LSX LC	110
Terms and Concepts	110
Metadata	110
Alternate Metadata	110
Result Set.....	110
Writeback Result Set	111
Token.....	111
LSX LC Classes	111
LCSession.....	111
LCCConnection.....	111
LCFieldlist.....	112
LCField.....	112
LCStream.....	112
LCNumeric	112
LCCurrency	112
LCDatetime	113
Working with the LSX	113
LSX LC Usage Notes	117
Environment	117
LSX LC Data Types	117
Standard Data types.....	117
Advanced Data types.....	118
Arrays and Indexes	118
Working with Multiple Rows of Data ..	119
Optional Parameters	119
Performance	120
Connector Caching	120

Tight Loops	120
NOTES.INI Variables	121

Chapter 2

LCCConnection Class..... 123

Overview	123
LCCConnection Properties	123
LCCConnection Class	
Methods Summary	124
Connection	124
Create Result Set	124
Data Manipulation.....	125
Metadata Manipulation	125
Miscellaneous.....	125
Connector Properties.....	125
New method for LCCConnection	126
Defined In	126
Syntax.....	126
Parameters.....	126
Usage Notes	126
Action method for LCCConnection	127
Call method for LCCConnection	129
Catalog method for LCCConnection	131
Connect method for LCCConnection.....	135
Copy method for LCCConnection	136
Create method for LCCConnection	138
Disconnect method for LCCConnection	141
Drop method for LCCConnection.....	142
Execute method for LCCConnection	144
Fetch method for LCCConnection	146
GetProperty method for LCCConnection	150

Contents

GetProperty<Type> Methods.....	152	New Constructor method for LCDatetime	194
Insert method for LCCConnection.....	157	Adjust method for LCDatetime.....	195
ListProperty method for LCCConnection	160	Clear method for LCDatetime.....	197
LookupProperty method for LCCConnection	163	Compare method for LCDatetime	198
Remove method for LCCConnection.....	165	Copy method for LCDatetime	200
Select method for LCCConnection	168	GetDiff method for LCDatetime	202
SetProperty method for LCCConnection	171	SetConstant method for LCDatetime	204
SetProperty<Type> methods for LCCConnection	173	SetCurrent method for LCDatetime	206
Update method for LCCConnection	176		
Chapter 3		Chapter 5	
LCCurrency Class.....	179	LCField Class.....	209
Overview	179	Overview.....	209
LCCurrency Class Methods Summary ..	180	LCField Class Methods Summary.....	209
LCCurrency Properties.....	180	Allocation.....	209
New method for LCCurrency.....	181	Field Properties	210
Add method for LCCurrency.....	182	Field Format.....	210
Compare method for LCCurrency.....	184	Field Data.....	210
Copy method for LCCurrency	186	Miscellaneous	211
Subtract method for LCCurrency	188	LCField Properties	212
		Field Format.....	214
Chapter 4		Number (INT, FLOAT, CURRENCY, NUMERIC).....	214
LCDatetime Class	191	Datetime (DATETIME).....	214
Overview	191	Stream (TEXT, BINARY)	215
Type DATETIME format and values....	191	Field Virtual Codes - Advanced Usage.	215
LCDatetime Class Methods Summary...	192	New method for LCField	216
LCDatetime Properties	193	ClearVirtualCode method for LCField	217
		Compare method for LCField	219
		Convert method for LCField.....	222
		Copy method for LCField	224

GetCurrency method for LCField.....	226
GetDatetime method for LCField.....	228
GetFieldlist method for LCField	230
GetFloat method for LCField	232
GetFormatDatetime method for LCField..	234
GetFormatNumber method for LCField...	236
GetFormatStream method for LCField.....	238
GetInt method for LCField	240
GetNumeric method for LCField.....	242
GetStream method for LCField	244
LookupVirtualCode method for LCField .	246
SetCurrency method for LCField	248
SetDatetime method for LCField.....	250
SetFieldlist method for LCField	252
SetFloat method for LCField.....	254
SetFormatDatetime method for LCField ..	256
SetFormatNumber method for LCField....	258
SetFormatStream method for LCField	260
SetInt method for LCField.....	262
SetNumeric method for LCField	264
SetStream method for LCField.....	266
SetVirtualCode method for LCField	268

Chapter 6 LCFieldlist Class..... 271

Overview	271
Accessing Field Data.....	271
Fieldlist Merging	272
Mapping and Merging	272
LCFieldlist Class Methods Summary	274
Creation.....	274
Fieldlist Retrieval	274
Fieldlist Modification.....	274
Fieldlist Merging	275
Fieldlist Mapping	275
LCFieldlist Properties	276
New method for LCFieldlist.....	277
Append method for LCFieldlist	278
Copy method for LCFieldlist	280
CopyField method for LCFieldlist	282
CopyRef method for LCFieldlist.....	284
GetField method for LCFieldlist	286
GetName method for LCFieldlist.....	288
IncludeField method for LCFieldlist.....	291
Insert method for LCFieldlist.....	293
List method for LCFieldlist.....	295
Lookup method for LCFieldlist.....	298
Map method for LCFieldlist.....	300
MapName method for LCFieldlist	303
Merge method for LCFieldlist.....	306

Contents

MergeVirtual method for LCFieldlist.....	310
Remove method for LCFieldlist	312
Replace method for LCFieldlist	314
SetName method for LCFieldlist.....	316

Chapter 7

LCNumeric Class 319

Overview	319
Type NUMERIC format and values	320
LCNumeric Properties.....	320
LCNumeric Class Methods Summary	321
New method for LCNumeric	322
Add method for LCNumeric	323
Compare method for LCNumeric.....	325
Copy method for LCNumeric.....	327
Subtract method for LCNumeric	329

Chapter 8

LCSession Class..... 331

Overview	331
Properties	331
LCSession Class Methods Summary	331
New method for LCSession.....	333
ClearStatus method for LCSession.....	334
GetStatus method for LCSession.....	336
GetStatusText method for LCSession	338
ListConnector method for LCSession	340

ListMetaConnector method for LCSession	346
LookupConnector method for LCSession	352
LookupMetaConnector method for LCSession	358
Sleep method for LCSession.....	364

Chapter 9

LCStream Class.....365

Overview	365
Type TEXT format and values.....	366
Type BINARY format and values.....	366
Stream flags.....	367
Stream format.....	367
Stream Buffer	368
LCStream Properties	369
New method for LCStream	370
Clear method for LCStream	371
Append method for LCStream	372
Compare method for LCStream	374
Convert method for LCStream.....	376
Copy method for LCStream	378
Extract method for LCStream	380
Merge method for LCStream	382
ResetFormat method for LCStream	384
SetFormat method for LCStream	386
Trim method for LCStream.....	388
DatetimeListGetRange method for	

LCStream	389	Error Message Components	413
DatetimeListGetValue method for LCStream	391	General Errors	414
DatetimeListInsertRange method for LCStream	393	Format of General Errors	414
DatetimeListInsertValue method for LCStream	395	General Errors	414
DatetimeListRemoveRange method for LCStream	397	Connector Errors	415
DatetimeListRemoveValue method for LCStream	399	Parameterized Connector Errors	422
NumberListGetRange method for LCStream	400	Field-Specific Parameterized Connector Errors	425
NumberListGetValue method for LCStream	402	Appendix B	
NumberListInsertRange method for LCStream	404	Connector Property Tokens....	429
NumberListInsertValue method for LCStream	405	Predefined Tokens.....	429
NumberListRemoveRange method for LCStream	406	Connector Identification	
NumberListRemoveValue method for LCStream	407	Property Tokens	431
TextListFetch method for LCStream.....	408	Appendix C	
TextListInsert method for LCStream.....	410	Connector Properties	433
TextListRemove method for LCStream....	411	Connector Properties.....	433
Appendix A		Notes Connector Properties	434
Error Messages	413	Notes Connector Property Combinations	441
Anatomy of an Error Message	413	Notes Virtual Fields	442
Example Error Message	413	DB2 Connector Properties	444
		EDA/SQL Connector Properties	447
		File System Connector Properties	449
		File System Metadata	451
		ODBC Connector Properties.....	452
		Oracle Connector Properties	454
		Sybase Connector Properties.....	457
		Appendix D	
		Character Sets	461
		List of Supported Character Sets.....	461

Preface

This manual provides information on how to set up Domino Connectors, how to utilize Domino Enterprise Connection Services (DECS) to access enterprise data in real-time, and also provides reference material for programming with the LotusScript Extension for Domino Connectors.

Structure of this manual

This manual consists of three parts:

Part 1 provides information on the Enterprise Connectivity Services.

Part 2 provides reference material for the LotusScript Extension for Lotus Connectors.

Part 3 provides information on Lotus Enterprise Integration Connectivity.

Structure of Notes and Domino documentation

Documentation for Notes and Domino is provided online in three databases available from the Help menu:

- Notes 5 Client Help
- Domino 5 Administration Help
- Domino 5 Designer Help

In addition, the Administration and Designer documentation is available as printed books. In Notes, select File - Other Help to see a table of all the available documentation. You can order books from the Lotus Web site at: www.lotus.com/store.

Documentation for the Notes Client

In addition to the online Help, the printed book *Notes Step by Step* provides a tutorial for beginning Notes users.

Documentation for Domino Administration

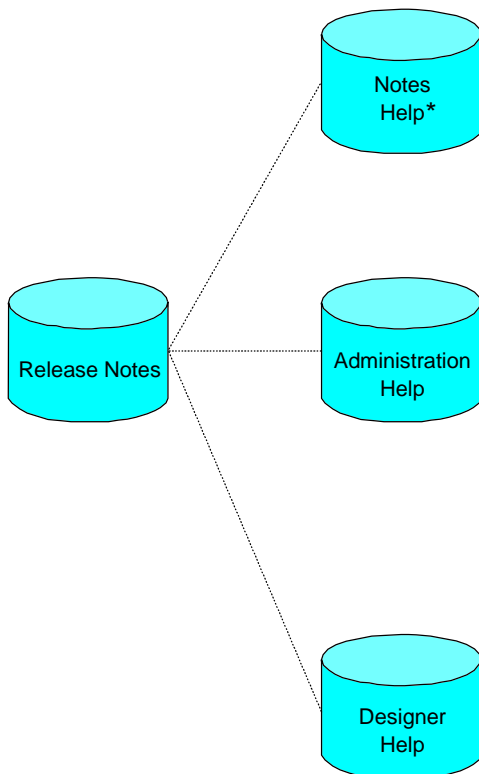
The following table shows the printed books that comprise the Domino Administration documentation set. The information in these books is also found in the Domino 5 Administration Help online database.

<i>Moving to Notes and Domino Release 5</i>	Describes how to upgrade existing Domino™ servers and Notes™ clients to Release 5. Also describes how to move users to Domino from other messaging systems.
<i>Configuring the Domino Network</i>	Explains how to configure a specific network to work with Domino. Also illustrates how to run Notes using multiple network protocols and individual protocols, such as AppleTalk, Banyan VINES, NetBIOS, Novell SPX (NetWare), and TCP/IP.
<i>Administering the Domino System, Volumes 1 and 2</i>	Describes how to set up and manage servers, users, server connections, mail, replication, security, calendars and scheduling, Web servers, NNTP services, billing, and system monitoring. Describes how to troubleshoot system problems.
<i>Administering Domino Clusters</i>	Describes how to set up, manage, and troubleshoot Domino clusters.
<i>Managing Domino Databases</i>	Provides information on managing databases, including putting databases into production, setting up access control lists and replication, and maintaining databases.

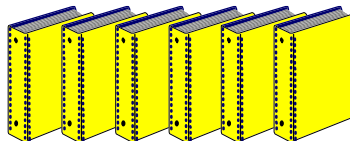
Documentation for Domino Designer

The following table shows the printed books that comprise the Domino Designer™ documentation set. The information in these books is also found in the Domino 5 Designer Help online database.

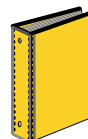
<i>Application Development with Domino Designer</i>	Explains how to create all the design elements used in building Domino applications, how to share information with other applications, and how to customize applications.
<i>Domino Designer Programming Guide Volume 1: Formula Language</i>	Introduces programming in Domino Designer and describes the formula language, the @functions, and the @commands.
<i>Domino Designer Programming Guide Volume 2: LotusScript Classes</i>	Provides reference information on the LotusScript® classes, which provide access to databases and other Domino structures.
<i>Domino Designer Programming Guide Volume 3: Java Classes</i>	Provides reference information on the Java classes, which provide access to databases and other Domino structures.
<i>LotusScript Language Guide</i>	Describes the basic building blocks of LotusScript, how to use the language to create applications, an overview of the LotusScript programming language, and a comprehensive list of language elements.
<i>Domino Enterprise Integration Guide</i>	Provides information on how to set up Domino Connectors, how to utilize Domino Enterprise Connection Services (DECS) to access enterprise data in real-time, and reference material for programming with the LotusScript Extension for Domino Connectors.
<i>Managing Domino Databases</i>	Provides information on managing databases, including putting databases into production, setting up access control lists and replication, and maintaining databases.



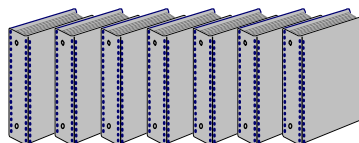
Step by Step**



Setting up a Domino Server
 Moving to Notes and Domino Release 5
 Configuring the Domino Network
 Administering Domino Clusters
 Administering the Domino System, Volume 1
 Administering the Domino System, Volume 2



Managing Domino Databases



Application Development with Domino Designer
 Domino Designer Templates Guide*
 Domino Designer Programming Guide, Volume 1: Formula Language
 Domino Designer Programming Guide, Volume 2: LotusScript Classes
 Domino Designer Programming Guide, Volume 3: Java Classes
 LotusScript Language Guide
 Domino Enterprise Integration Guide

Domino Objects
 Posters

* not available in print
 ** print only



Part 1: Domino Connectors

Setup Guide

Chapter 1

Introduction

This chapter provides information about this manual and related documentation.

Overview

This manual provides information related to enterprise connectivity for Domino Enterprise Connection Services, LotusScript Extension for Lotus Connectors, and Lotus Enterprise Integrator.

Included in this manual is information about the software required to connect to external data sources and to access the data in those external sources.

Also included in this manual is information about the Lotus Domino Connector connectivity test program, LCTEST. This program runs a connectivity test that establishes that the client data source access libraries are available and functional on the Domino Server. The test program does not test any specific functionality of the products, but ensures that the communications and client software required to access a specific data source is available and properly configured.

Organization of this Manual

This manual includes the following sections.

Section	Description
Chapter 1 Introduction	This chapter provides information about the organization of this manual and includes information about related documentation.
Chapter 2 LCTEST	This chapter provides information and instructions for running the connectivity test program, LCTEST.
Chapter 3 DB2 Connectivity	This chapter provides information and instructions for testing your system connectivity to DB2.
Chapter 4 EDA/SQL Connectivity	This chapter provides information and instructions for testing your system connectivity to EDA/SQL.
Chapter 5 ODBC Connectivity	This chapter provides information and instructions for testing your system connectivity to ODBC.
Chapter 6 Oracle Connectivity	This chapter provides information and instructions for testing your system connectivity to Oracle.
Chapter 7 Sybase Connectivity	This chapter provides information and instructions for testing your system connectivity to Sybase.

Related Documentation

Refer to the documents below for more information.

Domino Enterprise Connection Services

For information about Domino Enterprise Connection Services, refer to the *Domino Enterprise Connection Services User's Guide*, provided as an online NSF file (DECSDOC.NSF). You can also refer to the online field and popup help in the Connection Server documents.

LotusScript Extension for Lotus Connectors

- *LotusScript Extension for Lotus Connectors Reference Guide* - This manual, provided as an NSF file (LSXLCDOC.NSF), describes the LotusScript Extension for Lotus Connectors included with Domino 4.63 and above.

Enterprise Integrator Documentation

For more information about Enterprise Integrator and related products, refer to the following documents:

- *Enterprise Integrator 3.0 Release Notes* - The release notes contain information about the current release of Enterprise Integrator that may not be included in the printed documentation. We recommend that you read the release notes to review the information they contain.
- *Enterprise Integrator C API Reference Guide* - This is an online document delivered with the Enterprise Integrator software developer's kit (available on the Lotus Toolkit Collection CD)) and on the Lotus Enterprise Integration web location at www.eicentral.lotus.com. This may be used by developers wishing to create new Connectors for external data sources, which can then be operated via DECS, the Lotus Connector LSX and Lotus Enterprise Integrator.
- *Enterprise Integrator Java Classes Reference Guide* - This online, HTML document describes the Java classes that can be used to write Java Activities.

Other Documentation

For more information that you may find helpful, refer to the following documents:

- *Notes Administrator's Guide* - Provides information for configuring and administering a Notes installation.
- *LotusScript 3.1 Language Reference* - Provides information about writing LotusScript programs. This could be useful if you want to use the Lotus Connector LotusScript Extensions.

Chapter 2

LCTEST

This chapter provides information about LCTEST, a program provided for testing system and application connectivity to external data sources.

Overview

LCTEST is a connectivity test program provided with Domino Enterprise Connection Services (DECS). It is installed in the Domino program directory if you choose to install DECS.

Supported Data Sources

LCTEST tests system connectivity to the following supported data sources:

- DB2 (not supported on Windows NT/Alpha or Solaris Intel Edition)
- EDA/SQL (not supported on Windows NT/Alpha or Solaris Intel Edition)
- ODBC (Open Database Connectivity)
- Oracle
- Sybase

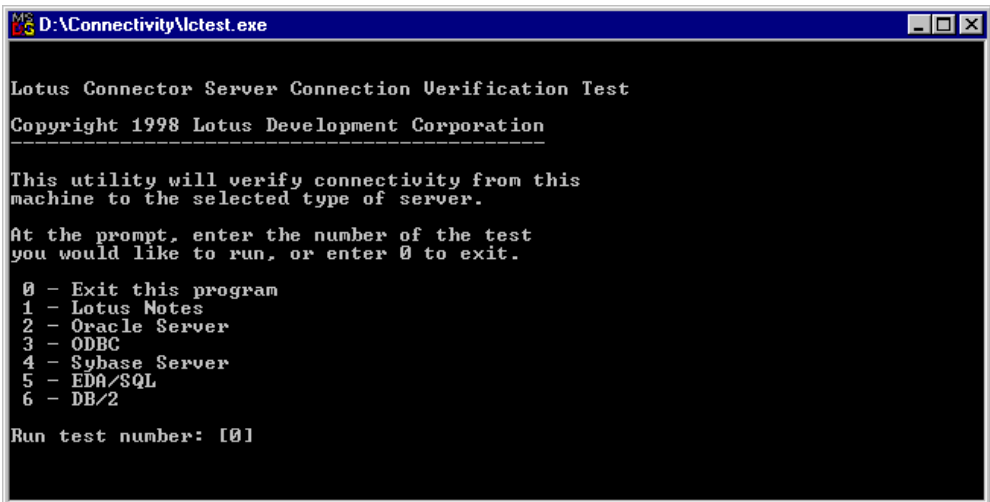
Requirements

Before running LCTEST, you must have the appropriate software installed on the Domino host for each data source you want to test. The remaining chapters of this manual provide information about the software required for each of the supported data sources.

Running LCTEST

Follow the steps below to run LCTEST.

1. Locate the LCTEST.EXE program specific to your operating system platform in the Domino program directory. The LCTEST program has the following names for each of the associated operating system platforms:
 - NLCTEST.EXE – for Windows 95, Windows NT (Win32)
 - ILCTEST.EXE – for OS/2
 - ALCTEST.EXE – for Windows NT/Alpha
2. Double-click on the program name to launch it, or type the program name at the system prompt. The LCTEST screen appears, as shown below.



```
D:\Connectivity\lctest.exe

Lotus Connector Server Connection Verification Test
Copyright 1998 Lotus Development Corporation
-----
This utility will verify connectivity from this
machine to the selected type of server.

At the prompt, enter the number of the test
you would like to run, or enter 0 to exit.

0 - Exit this program
1 - Lotus Notes
2 - Oracle Server
3 - ODBC
4 - Sybase Server
5 - EDA/SQL
6 - DB/2

Run test number: [0]
```

3. Enter the number of the test you want to run and press Enter. Depending on the type of data source you are testing, you are prompted to enter additional information as required to log in to the specified data source. For more information, refer to the chapter in this manual that discusses the specific data source for which you want to test connectivity.

Chapter 3

DB2 Connectivity

This chapter provides instructions and information for setting up and testing connectivity to DB2.

Requirements for DB2 Connectivity

Connectivity software requirements depend on the operating system platform and the specific version of DB2 that you are using.

- IBM eNetwork Communications Server for your operating system platform (see www.software.ibm.com/enetwork/commserver for more information)
- DB2 Connect Personal Edition
- DB2 Enterprise Edition

OR

- DB2 CAE (Client Application Enabler) version 2.1.2 or later.
- In addition, to connect to DB2 on an AS/400 or mainframe, a DDCCS gateway must be installed.

DB2 Connectivity Test

You should test for connectivity to the DB2 servers. To test for connectivity:

1. Run the version of the test program LCTEST, located in the Domino program directory, appropriate to your operating system. See Chapter 2 for more information.
2. Select DB2 from the program menu.
3. When the program prompts for a DB2 Database, User Name, and Password, enter valid connection information. The database must be cataloged in the DB2 database directory. (Refer to your DB2 Client documentation for further information on configuring a connection to a database.)
4. After entering the DB2 database, user name and password, the program attempts to connect to the DB2 Server. A message appears, telling if the test was successful or not.
5. You can retry a connection by entering Y at Try Again? [N]. This provides the opportunity to re-enter all of the required information, in case a mistake was made in spelling or you entered the wrong database, user name, or password.

Server Connectivity to DB2

This section provides information about software required to connect to DB2. This information is provided to help you get started. You should refer to the documentation for the specific software you are using for complete instructions.

IBM is currently shipping DB2 UDB Version 5. The specific product to use will depend on your environment.

Here is a brief listing of connectivity software available with DB2 Version 5:

- **DB2 Workgroup Edition:** Includes Client Pack CD for client connectivity, but does not include support for MVS/ESA, OS/390, OS/400, VM and VSE.
- **DB2 Enterprise Edition:** Includes all the functionality of DB2 WorkGroup Edition, plus support for host connectivity providing users with access to DB2 databases residing on host systems including MVS/ESA, OS/390, OS/400, VM and VSE.
- **DB2 Client Application Enabler:** Enables a client workstation to access the DB2 Server. Refer to DB 2 documentation for supported platforms.
- **DB2 Universal Database Personal Edition:** Formerly know as DB2 Single Server. Enables you to create and use local databases, and to access remote DB2 databases. Available for OS/2 Win 95 and WINNT.
- **DB2 Connect Enterprise Edition:** Formerly know as DDCS Multi-User gateway. Provides access from clients on the network to DB2 databases that reside on hosts such as MVS/ESA, OS/390, OS/400, VM and VSE.
- **DB2 Connect Personal Edition:** Formerly know as DDCS Single-User. Provides access from a single workstation to DB2 databases residing on hosts such as MVS/ESA, OS/390, OS/400, VM and VSE. This product is only available for OS/2, Windows 95 and Windows NT.

Refer to the documentation provided with IBM DB2 Universal Database (the manual entitled "Road Map to DB2 Programming", Appendix A, "About DB2 Universal Database").

Connectivity to DB2 on AS/400

Direct DB2/400 Connectivity

The DB2 Connect Personal Edition is the only required DB2 software for connectivity. There is no need for a DB2 Client such as DB2 CAE.

1. One Server connecting to DB2/400

Install IBM DB2 Connect Personal Edition Version 5 on the Server machine.

Bundled with Connect Personal Edition is an SNA Server. Install the SNA Server with the SNA over APPC option. It must be over APPC; the AS/400 requires APPC for DB2 connectivity. Also, in this stand alone environment the SNA over APPC is self sufficient; there is no general network need for APPC and SNA in the network.

Refer to the IBM DB2 Connect Personal Edition Quick Beginnings Version 5 document (numbered S10J-8162-00) as a means to begin the installation. It states to use a user name and password that are a user name and password on the DB2/400. This will allow the installation to perform operations on the AS/400 without intervention.

This option does not require the installation of software to the AS/400.

2. Multiple Servers connecting to DB2/400

Install IBM DB2 Connect Enterprise Edition Version 5.

This will install a gateway to DB2/400. A gateway allows access to the DB2/400 data from multiple Servers. This software is only necessary if you plan to have more than one Server.

The Connect Enterprise Edition does not come with bundled SNA software, so it is up the user to install an SNA Server, such as Microsoft's SNA Server or the OS/2 Communication Manager software.

If the DB2 Connect Enterprise Edition is not installed on the machine, then a DB2 Client such as DB2 CAE needs to be installed onto the machine. This client is needed to perform the direct connection from the Domino Server to DB2.

This option does not require any software installed to the AS/400.

3. Server connecting to DB2/400 with DDCS

- Requirements: Installed DDCS and SNA Server.

This option is for customers with an existing DDCS installation. This option only exists for customers who have DDCS already in house, because at this time only the latest DB2 Connect Version 5 software is available for purchase.

DDCS Single User or Gateway can be used to connect to DB2/400. With DDCS, there is also a requirement for an SNA server over APPC. Therefore, you will need SNA software such as Microsoft SNA or OS/2 Communication Manager.

If DDCS is not installed on the Server machine, then a DB2 Client such as DB2 CAE needs to be installed on the Server machine. This client is needed to perform the direct connection from the Server to DB2/400.

This option does not require any software installed to the AS/400.

ODBC DB2/400 Connectivity

Several software packages support connectivity to DB2/400 through ODBC. The preferred approach to connectivity is through the DB2 Client. This gives the Server the advantage of straight connectivity, speed and datatype access without the ODBC layer. However, should a user prefer to install such products as Rhumba or IBM's Client Access for AS/400, then the connectivity would be through the ODBC Link.

Be certain to verify the requirements of the individual packages, in most cases an SNA server is still a requirement.

This option may or may not require additional software installed to the AS/400.

Connection for DB2 CAE and DDCS

DECS offer native connectivity to DB2. This provides a direct connection to DB2 on all DB2 supported platforms, and enhances the DB2 connectivity previously provided through ODBC. The native interface offers improvements in speed and support for native datatypes not accessible through ODBC. This section contains information on configuring native DB2 connectivity through DB2 CAE and DDCS.

Getting Started

Network communications programs must be installed to the Server and the DB2 system to establish a network connection for data transfer. Depending on the platform you are using to operate the Server, and the operating system used to store DB2, your requirements are either to install CAE/2 (Client Application Enabler/2) or DDCS (Distributed Database Connection Services). These communications products are available from IBM. The machines must have connectivity through the DB2 CAE (Client Application Enabler) or other DB2 run-time environment (i.e., DB2 Server). CAE/2 must be version 2.1.0 or above and must be native for the operating system it is running on (i.e., on OS/2, you must have CAE for OS/2 Warp; there is a separate version available for Windows NT).

For DDCS or Client Application Enabler OS/2 systems to access DRDA Application Servers such as DB2 for MVS/ESA, DB2 for VSE and VM, and DB2 for OS/400, you must also have the APPC protocol support installed. Instructions to do this are included below in the section “Communications Manager/2 Setup for DB2/2 to DB2/400 Server”.

For DDCS for Windows NT to access DRDA Application Servers such as DB2 for MVS/ESA, DB2 for VSE and VM, and DB2 for OS/400, the APPC protocol support must also be installed on your system. The program required to do this is Microsoft SNA Server version 2.11 or later.

Before attempting to connect, verify connectivity through the DB2 Command Line program (supplied with DB2) or the LCTEST connectivity test program, described at the start of this section.

Sample Steps for DB2 Configuration

The following illustrates some sample steps you may wish to follow when configuring your DB2 connectivity. For full details, refer to the DB2 manual "Install/Use DB2 Clients for xxx" where xxx is your operating system. The steps below assume you are installing the CAE for the Server machine running on OS/2, Windows NT or HP-UX, and are using TCP/IP. Note that you must install CAE Version 2.1 or above. The steps will be much the same if you instead install the DB2 Server on this machine.

1. Install the DB2 CAE software.
2. Ensure the Server machine can resolve the DB2 Server TCP/IP host address (i.e., you should be able to ping the server). If it can't, you need to either have the Domain Name Server updated to include the DB2 Server name and address or add an entry to your Server machine's hosts file. For example, on WARP this would consist of running the TCP/IP Configuration Utility and adding the DB2 Server Name and IP address in the Hostnames dialog. On Windows-NT and HP-UX, you must manually edit the hosts file (NT: c:\winnt35\system32\drivers\etc\hosts, HP-UX: /etc/hosts) and add an entry such as 9.21.15.235 tcpghost
3. Ensure the DB2 Server has enabled the TCP/IP protocol through the DB2COMM environment variable. This variable may indicate multiple protocols. Make certain it includes "TCPIP". This variable must be set at the time the DB2 Server is started.
4. Ensure the services file on the DB2 Server machine contains an entry for TCP/IP support for each database manager instance you plan on accessing. A second entry is required to support TCP/IP interrupt from DB2 V1.x Client and is not required if all your clients are V2.0 or above.

```
db2instlc  3700/tcp    # DB2 connection service port for V1 and V2.

                # Also serves as an interrupt

                # connection service port for DB2 V2.

db2instli  3701/tcp    # DB2 interrupt connection service port

                # for V1.x client releases
```

where db2instlc is the value of the service_name parameter, db2instli is arbitrary. 3700 and 3701 are the port numbers for the connection and interrupt port, and TCP is the protocol. The port number 3700 is arbitrary, but must be unique within the file. The second port number must also be unique, and equal to the first number plus one. These same numbers must be used when configuring the services file on the Server machine (step 6).

Connection for DB2 CAE and DDCS

5. On the DB2 Server, ensure the database manager is listening for connections for the DB2 instance. This is done by issuing the following command from the DB2 command line processor at the server:

```
UPDATE DATABASE MANAGER CONFIGURATION USING SVCENAME db2inst1c
```

where db2inst1c is the service name.

For the changes to take effect, restart the database manager at the server (issue db2stop and db2start in succession at the server).

NOTE: The svcename used must match the service name configured in the services file on both the client and the server.

6. Ensure the services file on the Server machine contains an entry matching the entry on the DB2 Server. Depending on the version of DB2 server you are connecting to, you need one or two entries in the services file:

```
db2inst1c    3700/tcp    # DB2 connection service port for V1 and V2.
               # Also serves as an interrupt
               # connection service port for DB2 V2.
db2inst1i    3701/tcp    # DB2 interrupt connection service port
               # for V1.x client releases
```

NOTE: You only need the first entry if you are connecting to a DB2 Version 2 server. This must match the service name entry in the server's services file.

You need both entries if you are connecting to a Version 1 DB2 server. These entries must match the corresponding server entries. The service name, port number (3700 and 3701 in our example) and protocol must be identical.

7. On the Server machine, catalog the DB2 Server and Database. To catalog the server, use the DB2 command line processor on the Server machine and enter the command:

```
CATALOG TCPIP NODE nodename REMOTE hostname SERVER servicename
```

where nodename is a name you pick to refer to this connection, hostname is the TCP/IP name of the DB2 Server machine, and servicename is the instance name you entered in the services file (you only need to do this once using the first port even if you also entered an interrupt connection service port in the services file).

Next, catalog the database with the command:

```
CATALOG DATABASE databasename AS local_database_alias AT NODE nodename
```

where `databasename` is the name of the database on the DB2 Server, `local_database_alias` is a name you pick which you will use to connect to the database from the Server machine, and `nodename` is the name you used in the previous CATALOG TCPIP command.

Exit and restart the DB2 command line processor. Try connecting to the DB2 database with the command:

```
CONNECT TO local_database_alias USER username
```

where `local_database_alias` is the alias you cataloged and `username` is a valid DB2 username.

You may now verify that will have connectivity to DB2 by running the `LCTEST.EXE` program found in the Domino program directory. This program will prompt you for the database name (`local_database_alias`), `userid`, and `password`.

When to use IBM Distributed Database Connection Services (DDCS)

DDCS has been replaced by DB2 Connect Personal Edition and DB2 Connect Enterprise Edition. The following information is provided for environments which may not yet have acquired either of these new connectivity packages.

When connecting to DB2 for MVS/ESA, DB2 for VSE and VM(SQL/DS), or DB2 for OS/400, you can use DDCS. DDCS can also be used to connect to any other DB2 server (e.g., WIN-NT, AIX, etc.), but it is more direct and efficient to go from the local machine using CAE directly to the DB2 server.

It does not matter where DDCS is installed as long as the Domino Server machine can connect to the DDCS machine through TCP/IP, SPX or any other DB2 supported protocol.

Refer to the DDCS Install/Configuration document for instructions and software requirements. Connections to external systems may require additional communications software. For example, to connect DDCS on OS/2 to MVS or AS400, the OS/2 machine must have APPC connectivity, namely IBM Communications Manager/2 Version 1.1 or later or IBM Communications Server for OS/2 Warp Version 4.

Sample Steps for using DDCS

As an example of how DDCS would work with the Server, let's assume the following:

The Server is running on an Windows NT machine called NP1. This machine has DB2 CAE for Windows NT installed and will connect to the DDCS workstation using TCP/IP. DDCS is installed on an OS/2 Warp machine called DB2GW. It also has TCP/IP and IBM Communications Server installed.

Connectivity to a DB2 database on the MVS machine MVS1 is desired and will be made through APPC.

1. The connectivity between DB2GW and MVS1 must be established through DB2GW's IBM Communications Server and MVS1's VTAM. The details of this configuration and configuration at the MVS host are not covered here, refer to the DDCS Install/Configuration manual for information.
2. MVS1's node and database is cataloged at the DDCS machine. The following commands are issued to catalog the remote node and database:

```
CATALOG APPC NODE db2node REMOTE db2pic SECURITY PROGRAM
```

where db2node is a name you pick to refer to this host, db2pic is the Symbolic Destination Name you defined when you configured MVS1 in the IBM Communications Server.

```
CATALOG DATABASE db2db AS mydb AT NODE db2node AUTHENTICATION DCS
```

where db2db is the MVS database name, mydb is the database alias, and db2node is the node defined in the previous command.

3. The services file on the DDCS machine (DB2GW) is modified to include an entry for a database instance (inst1c) and an interrupt connection (inst1i). The following DB2 command must be used to tell the database manager to listen for connections to the instance from remote clients to the instance.

```
UPDATE DATABASE MANAGER CONFIGURATION USING SVCENAME inst1c
```

The environment variable DB2COMM is set to TCPIP. DDCS is started with the operating system command DB2START.

4. Connectivity from NP1 is established through the DB2 CAE by defining the target database:

```
CATALOG TCPIP NODE gw1 REMOTE db2gw SERVER inst1c
```

```
CATALOG DATABASE mydb AS mvbdb2 AT NODE gw1
```

where gw1 is the alias that will refer to the DDCS Gateway and will be used by the Server, db2gw is the TCP/IP name of the DDCS Gateway machine, and inst1c is the service name defined in the step 3 above. mydb is the alias defined in step 2 above, mvbdb2 is the alias that will be used by DECS.

MDI Gateway

DECS includes built-in support for the use of an MDI Gateway with DB2. MDI Gateway is a Sybase product that enables Sybase clients to access DB2 data.

Chapter 4

EDA/SQL Connectivity

This chapter provides information about setting up connectivity to an EDA/SQL data source.

Requirements for EDA/SQL Connectivity

- The EDA/Client software for the host operating system. The EDA/Client version must be Release 3.2 or later and must be 32-bit on Windows NT and OS/2.
- An EDA Server on the platform where the EDA supported database resides.
- Connectivity to the EDA server.

EDA/SQL Connectivity Test

The EDA connectivity test checks connectivity between EDA and the database. To test for connectivity:

1. Run the version of the test program LCTEST, located in the Domino program directory, appropriate to your operating system. See Chapter 2 for more information.
2. Select EDA/SQL from the program menu.
3. Enter the EDA Server, user name, and password as prompted. The program then attempts to connect to the EDA data source.
4. A message appears indicating whether the test was successful or not.
5. You can retry a connection by entering Y at Try Again? This provides the opportunity to re-enter all of the required information, in case a mistake was made in spelling or you gave the wrong User Name, Password or EDA Server.

Chapter 5

ODBC Connectivity

This chapter provides information about setting up connectivity an ODBC data source.

Requirements for ODBC Connectivity

- The ODBC driver appropriate to the operating system.
- The driver must be 32-bit on NT and OS/2.
- The ODBC driver must be thread-safe.
- The ODBC Administrator must be present.
- There must be correctly defined ODBC data sources in that ODBC Administrator.

ODBC Connectivity Test

The ODBC connectivity test checks connectivity between ODBC and the database. To test for connectivity complete the following steps:

1. Run the version of the test program LCTEST, located in the Domino program directory, appropriate to your operating system. See Chapter 2 for more information.
2. Select ODBC from the program menu.
3. Enter the data source, user name, and password when the program prompts for them.
4. Choose (Y/N) whether or not you want detailed driver information.
5. You can produce a printed report for diagnostic purposes. When asked, you can choose to output to a file (Y) or not (N). If you do not choose output to a file, the results appear on your monitor.
6. If you chose file output, supply a name for the file, then press Enter. The program then attempts to connect to the ODBC data source.
7. A message appears telling whether the test was successful or not.
8. You can retry a connection by entering Y at Another Data Source?.

Chapter 6

Oracle Connectivity

This chapter provides information about setting up connectivity to an Oracle Server.

Requirements for Oracle Connectivity

- With an OS/2-based Server: Oracle SQL*Net version 2.
- With a Windows NT-based Server: Oracle SQL*Net version 1 or 2.
- In either case, SQL*Net must be same version as SQL*Net installed on the Oracle data server. A network connection must exist between the Server machine and the Oracle data server machine via SQL*Net.
- Native Oracle connectivity support requires Oracle version 7.2 or later.
- OS/2 works only with Oracle 7.3.
- Oracle Version 7.3 and HP-UX: You must obtain the Oracle Fix for Bug #441647. This applies patches to Oracle's libclntsh.sl.
- Oracle 8: Enterprise Integrator and DECS link with Oracle 7.3 libraries, which use SQL*NET for the communications layer. If you are using Oracle 8 with DECS or Enterprise Integrator, you must install Oracle SQL*NET. You can use SQLNET Easy Config utility to configure SQL*NET.
- When connecting from an Oracle 8 client to an Oracle 8 server, use of Oracle SQL*NET may be required. If you encounter access problems, install SQL*NET.

Oracle Connectivity Test

You should test for connectivity between the Domino Server and the Oracle servers. Complete the following steps to test connectivity:

1. Run the version of the test program LCTEST, located in the Domino program directory, appropriate to your operating system. See Chapter 2 for more information.
2. Select Oracle Server from the program menu.
3. When the program prompts for an Oracle User Name, Password, and Connection String, enter a valid Username and Password.

The Connection String can be for either SQL*Net V1 or SQL*Net V2, depending on what software you have configured on your Oracle Server and Domino Server.

The general format for a V1 string is `network_prefix:server_name:sid`.

The format for V2 consists of a single identifier, `service_name`.

Refer to your Oracle SQL*Net documentation for further information on the format of Connection Strings.

4. After entering a User Name, Password, and Connection String, the program attempts to connect to the Oracle Server.
5. A message appears telling whether the test was successful or not.
6. You can retry a connection by entering Y at Try Again? [N]. This provides the opportunity to re-enter all of the required information, in case a mistake was made in spelling or you gave the wrong Username, Password or Connection String.

Chapter 7

Sybase Connectivity

This chapter provides information about setting up connectivity a Sybase Server.

Requirements for Sybase Connectivity

- With an OS/2-based or Windows NT-based: System 10 Netlib.
- A network connection must exist between the Domino Server and the Sybase SQL Server via Netlib.

Sybase Connectivity Test

You should test for connectivity between the Domino Server and the Sybase SQL Server servers. Complete the following steps to test for connectivity:

1. Run the version of the test program LCTEST, located in the Domino program directory, appropriate to your operating system. See Chapter 2 for more information.
2. Select Sybase Server from the program menu.
3. Enter the server name, user name, and password as prompted. The test program then attempts to connect to the Sybase SQL Server machine.
4. A message appears telling whether the test was successful or not.
5. You can retry a connection by entering Y at Try Again? [N]. This provides the opportunity to re-enter all of the required information, in case a mistake was made in spelling or you gave the wrong user name, password or server name.
6. If problems persist, check to make sure you have the client software properly installed.



Part 2: Domino Enterprise Connection Services

User's Guide

Chapter 1

Introduction

This chapter provides information about the organization of this manual. This chapter also includes an overview of the Domino Enterprise Connection Services (DECS).

Organization of this Manual

This manual includes the sections described below.

Section	Description
Chapter 1 Introduction	This chapter provides information about the organization of this manual, and includes an introduction to the Domino Enterprise Connection Services and an overview of its functionality.
Chapter 2 DECS Administrator	This chapter provides information about the DECS navigator, views, and menu commands.
Chapter 3 Defining Data Sources	This chapter describes how to define external data sources.
Chapter 4 Creating RealTime Activities	This chapter describes how to create RealTime Activities using the RealTime Activities wizard.
Chapter 5 Building the Notes Application	This chapter provides information for creating a Notes application for use with a RealTime Activity.
Chapter 6 Examples Using RealTime Options	This chapter provides examples that show how to use RealTime options to refine the results of RealTime Activities.
Chapter 7 RealTime Dynamic Queries	This chapter provides information on how to use RealTime Activities for on-demand, dynamic queries of external data.

Introduction to Domino Enterprise Connection Services

The Domino Enterprise Connection Server enables you to create RealTime Activities. A RealTime Activity provides synchronous access from a Domino application to a supported external data source.

The RealTime Activity intercepts Notes database events. For example, when Notes or web client users open, create, update, or save Notes documents, these events are intercepted and acted upon, obtaining "real-time" access from the Notes form to external data sources supported by the Domino Enterprise Connection Server. Real-time means that you get the data immediately, relative to the network bandwidth and other processes running that may affect system performance.

Once a system administrator has created the RealTime Activity, identifying a particular form within a Domino application to have certain fields populated by an external database source, Notes users can open, create, update or delete external, backend data directly and transparently through their familiar Notes Client. By extension, web clients may open the same Notes forms by accessing a Domino Release 4.6 or greater server, and obtain RealTime access to supported external source data.

The Domino Enterprise Connection Server, running on the Domino Server that is hosting the Notes application, intercepts and handles the Notes database events.

For example, if the external database to be queried or updated from the Notes form is DB2, Notes end-users may work with DB2 data as if it were in Notes. DB2 connectivity software is not required on the client system. Network access to the external data source is handled by the Domino server machine, which contains the connectivity software for the external data source, such as DB2. No programming is required to accomplish this functionality. In addition, it is an option to store retrieved data to the Notes form, or to simply view the retrieved data, potentially reducing storage requirements on the Notes side.

When creating a RealTime Activity, several items are required to provide RealTime data access from a single Notes form. Each RealTime Activity monitors a specific Domino application form and requires a Notes Form to define the metadata. Metadata is the list of fields in the Notes form and a list of fields in the external data source that are mapped during data query or update from the Notes application. Within an application, one or multiple external sources may be accessed from the Notes form. A single external data source definition indicates the data source to connect to and the metadata to use. Key and field mappings are also required. Several RealTime Activities can monitor different databases, a single database or even a single form. This means a single document can be populated real-time, consisting of data from multiple backend databases using a RealTime Activity for each of the various backend data sources.

Supported Data Sources

RealTime Activities support a range of external data sources, called Connectors. Additional data sources are continuously being added. As Lotus and other third parties create data source Connectors for Domino, they will all be manageable using the Domino Enterprise Connection Services product. Currently supported data sources include:

- DB2
- EDA/SQL
- Open Database Connectivity (ODBC)
- Oracle
- Sybase

Each of the data sources has specific software requirements for network connectivity that must be met by the Domino Server. Refer to the *Domino Connectors Setup Guide* for information about the connectivity requirements for each of the supported data sources.

Contact and Support Information

Lotus provides extensive support for its products. The following sections describe the different ways in which you can get help on using RealTime Activities, as well as information on how to contact us with suggestions and recommendations.

Enterprise Integration Sales Support

You can reach the Lotus Enterprise Integration Business Unit Sales staff at:

1-800-944-7358

Telephone Support

You can reach the Technical Support group at:

1-800-346-6388

Contacting 3rd Party Support

In some cases, you may need to contact customer support for another program or application in your environment. These could include the vendors of any databases you may be using, such as Sybase or Oracle. Refer to the documentation for the specific database or application for more information.

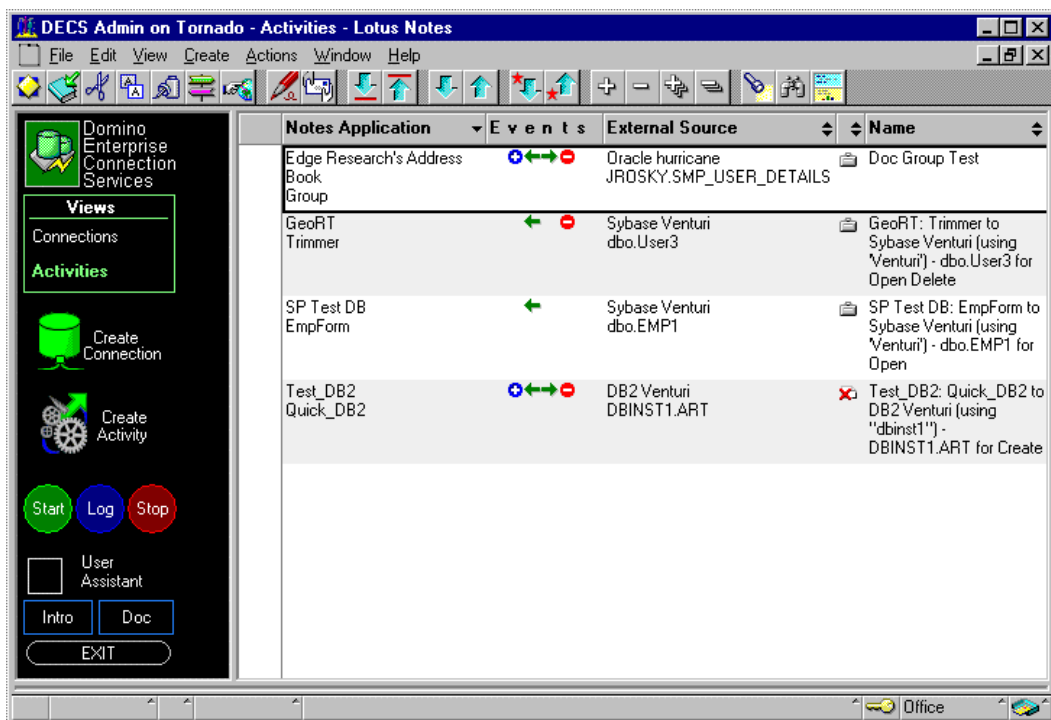
Chapter 2

DECS Administrator

This chapter provides information about the DECS Administrator. Included is information about the navigator, views and menu commands.










The DECS Administrator

Shown below is the DECS administrator. The navigator, views and menu commands in the administrator are described in the sections that follow.



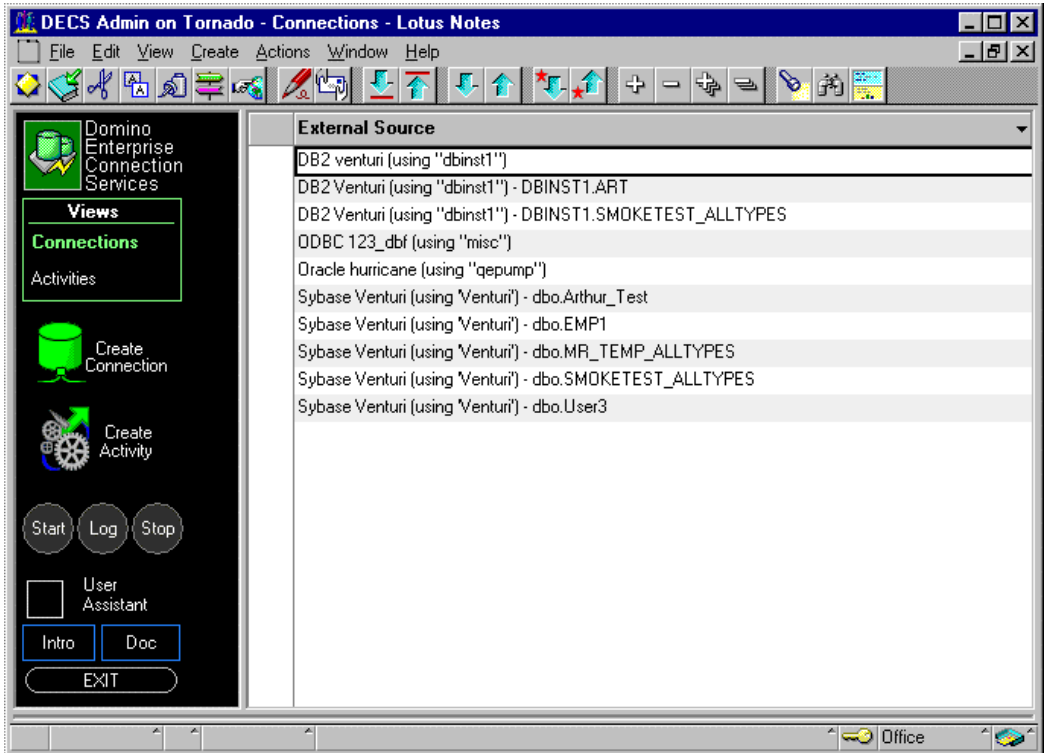
DECS Navigator

The table below gives a description of the commands available in the DECS navigator.

	Selects a view of available Connections or a view of RealTime Activities.
	Creates a new Lotus Connection document for an external data source. This launches a wizard that prompts you through the process of defining a Connection to an external data source.
	Creates a new RealTime Activity. When the User Assistant is active, this launches a wizard that prompts you through the process of defining a RealTime Activity between your Notes application and your external data source. When the User Assistant is turned off, this displays a blank RealTime Activity document that you can edit directly.
	Begins execution of the currently selected RealTime Activity. This has no effect if the current selection is already executing. This is disabled when in Connection view.
	Displays the status of the currently selected RealTime Activity. If the current selection is running, this will display the start time and other current status. If the current selection is not running, this will display the results of the most recent execution. This is disabled when in Connections view.
	Ends execution of the currently selected RealTime Activity. This has no effect if the current selection is not running. This is disabled when in Connections view.
	Toggles the User Assistant. When turned on, this provides additional help and enables the New Activity wizard. This is useful for first time and infrequent users. The New Activity wizard guides you through creating the RealTime Activity document and provides information to assist in the creation and editing of the document.
	Displays this page of information.
	Displays the online documentation.
	Closes the DECS administrator.

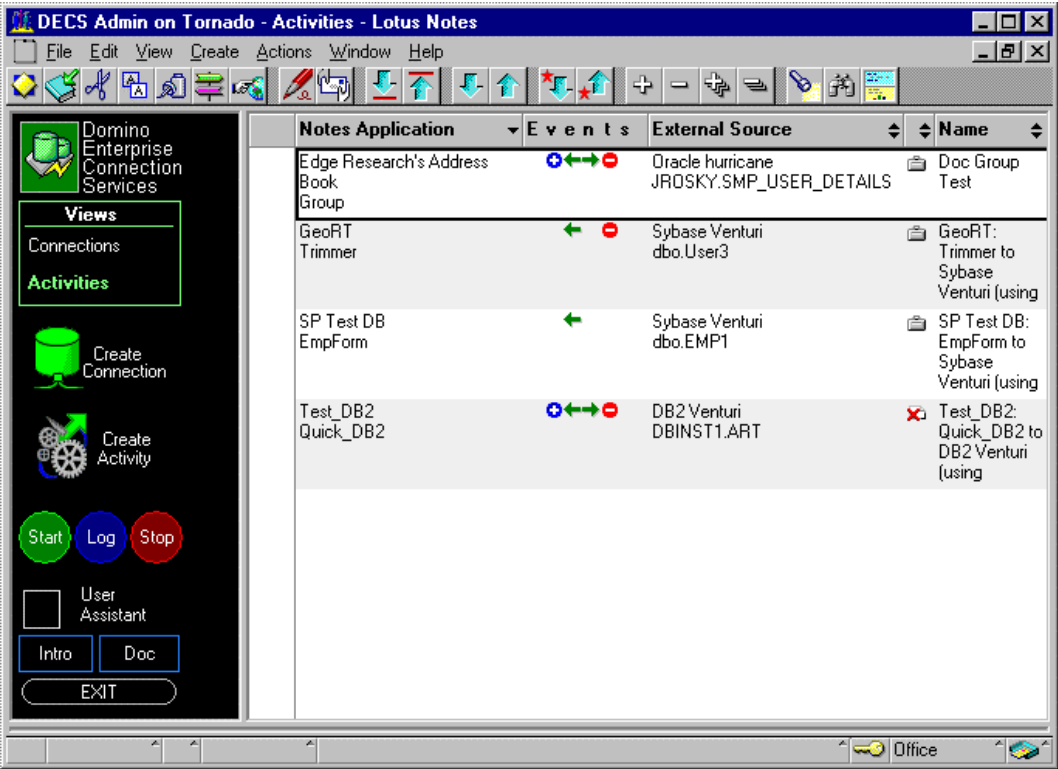
Connections View

Shown below is an example of the Connections view. Lotus Connection documents are named automatically by the DECS task using a convention that specifies the external system, the database, and the table name. When applicable, the user name required for login to the data source is shown in parentheses.














RealTime Activities View

Shown below is an example of the RealTime Activities view.



The table below describes the information shown in the Connections view of the RealTime Activity administrator.

Notes Application		Shows the name of the Notes NSF and the specific form being monitored.
Events		Indicates the Activity will process document creations.
		Indicates the Activity will process document opens.
		Indicates the Activity will process document updates (edit/save).
		Indicates the Activity will process document deletions.
External Source		Shows the type of external data source, the database and table name or metadata.
Status		Shows the status of the RealTime Activity, as indicated by the icons below. Refresh the view periodically to update the status.
		New
		Active
		Starting / Stopping
		Stopped with an error
		Disabled with an error
		Scheduled to AutoStart with server
		Not scheduled
Name	The user-specified name of the Activity.	

Menu Commands

The following commands are available from the **Actions•Tools** menu:

Command	Description
Reset Connection	Restarts the currently selected RealTime Activity.
Initialize Keys	<p>Populates the currently selected RealTime Activity key fields with data from the external data source for that connection. This command should be used once after creating a RealTime Activity. Running it again creates duplicates of existing documents.</p> <p>If you have specified <i>Leave Selected RealTime Fields in Document</i> or <i>Leave All RealTime Fields in Document</i>, the associated fields are transferred to the Notes application.</p> <p>NOTE: When using Initialize Keys with a RealTime Activity that has a Filter Formula, you are asked if you want to use a conditional clause to restrict the number of external records imported into Notes. If you choose Yes, you are prompted to enter the external conditional clause. The conditional clause must be in the external system's syntax for a keyed selection clause. For SQL based systems, the conditional clause would be the portion of a WHERE clause following the WHERE. For example, if the complete statement were: <code>SELECT * from Table WHERE Number > 1</code>, you would only enter <code>Number > 1</code> as the key initialization condition.</p>

Chapter 3

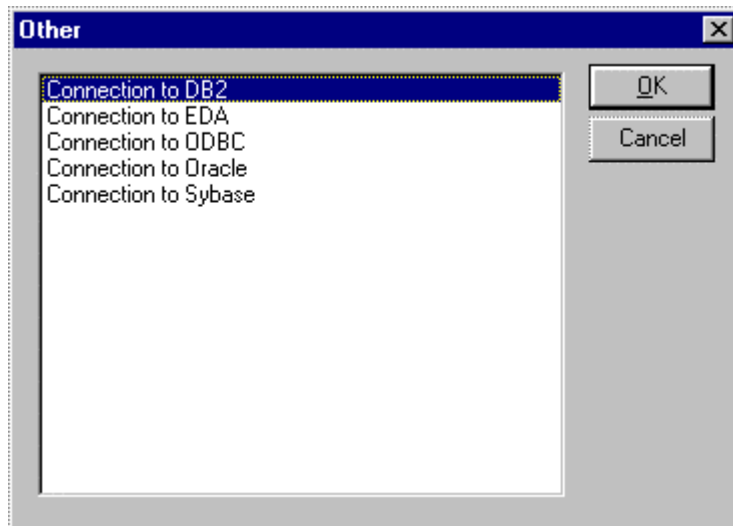
Defining Lotus Connections

This chapter provides information and instructions for creating Lotus Connection documents.

Defining Lotus Connections Using the Wizard

You use the Lotus Connection document wizard to define connections to external data sources.

1. Click on the Create Connection icon - . A list of supported data sources appears, as shown below.



NOTE: In order to connect to a supported data source, you must have the required connectivity software for that data source installed on your server.

You can remove any of the supported data sources from the list that you do not intend to use. Refer to the *Notes Application Developer's Guide* for more

Defining Lotus Connections Using the Wizard

information on how to remove a form from a menu.

2. Select the data source for which you want to define a connection from the list of available sources and click OK. The Lotus Connection document appears. Below is an example of a Lotus Connection for Oracle document. Note that the text in the top part of the document only appears when the User Assistant is turned on in the DECS navigator.

New Connection to - Lotus Notes

File Edit View Create Actions Text Window Help

Save and Close Author Privileges

Connection to Oracle

Edward Knowlton/CAM/Lotus

- 1 Connectivity Parameters**
Fill out the Host String specifying which Oracle system you wish to access. Then specify the user name and password to use to log onto the selected Oracle system.
- 2 Table Selection**
Click on the down-arrow button to view a list of tables in the selected data source. When you select one of the tables, all of the columns in that table are automatically inserted here. This information is useful when developing the Notes application for use with this Lotus Connection.
- 3 What's Next ...**
After completing this Connection document, you are ready to build your Notes application that will monitor this data source. You may use this Connection document at anytime as a reference to review the fields and data types available in this data source. Once you have completed your Notes application, you may create a RealTime Activity. During this process, you will be prompted to select a Lotus Connection to use. Choosing this Connection will connect your Notes application to this Lotus Connection.

Connectivity

Host String:

User Name:

Password: Password(s) NOT Encrypted

Selection Type: ☒ Table ☐ View

Table Selection

Owner: <Any>

Name:

Column(s):

Comment:





Override

Enter the Oracle Database Server.

Office

Defining Lotus Connections Using the Wizard

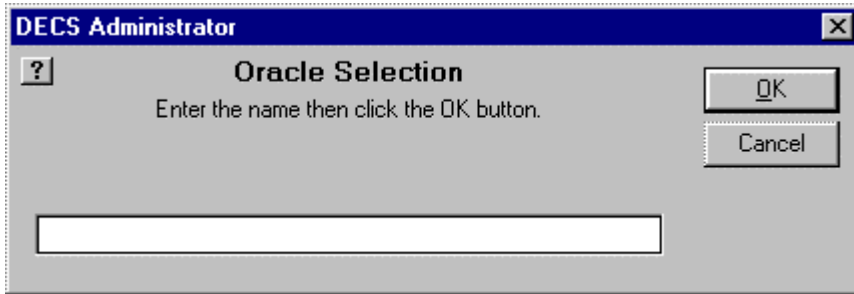
- Enter the required information in the Lotus Connection document. The table below describes the fields in the Lotus Connection document.

Connectivity	This section of the document specifies the database or data source and the connectivity information required to access that data source.
Database/Host String/SQL Server:	Enter the information required for the specific data source for which you are defining a connection. This information can vary according to the type of data source selected.
User Name:	Enter the User Name required to access the selected data source specified above.
Password:	Enter the Password associated with the User Name specified above.
 - Password Encryption Key	Click on the Password encryption icon to encrypt your password for this Lotus Connection document. You can click it again to toggle encryption off.
Selection Type:	You can choose to connect to data in tables or views. Select the desired option.
Table Selection	Use this section of the document to select the specific metadata that you want to access through this Connection.
	The Override button is provided for situations in which DECS is unable to browse all of the metadata due to size limitations. You can use the Override button to select an owner and data source that do not appear in the Name list of data sources.. See the section below, "Using the Override Button," for more information.
Owner: 	This option allows you to select from tables by owner name. The default value is <Any>, allowing you to select from all tables in the database. Click on the down-arrow button to see a list of existing owner names. This enables you to select only from tables belonging to the specified owner.
Name: 	This field shows the name of the specific table or view you select from the data source. Click on the down-arrow button to access a list of the tables or views from which you can choose.
Column(s):	This field lists the column names and their associated data types found in the selected table or view of the external data source.
Comments	This field allows you to enter text that may be helpful to you in organizing or keeping track of your Connectivity documents.

Using the Override Button

You use the Override button to specify an owner and a data source that are not included in the list of tables or views shown when you select the down-arrow button next to the Name field.

You can enter the owner name and the table name, or just the table name. You must enter these items exactly as they are named in the data source or DECS will not be able to locate them.



Example of a Completed Lotus Connection document

Below is an example of a completed Lotus Connection document for a Sybase database.

Sybase Venturi (using 'Venturi') - dbo.SMOKETEST_ALLTYPES - Lotus Notes

File Edit View Create Actions Window Help

Edit Document

Connection to Sybase /CAM/Lotus

Connectivity

SQL Server:	bedlam
Database:	Venturi
User Name:	Venturi
Password:	Venturi Password(s) NOT Encrypted
Selection Type:	<input checked="" type="radio"/> Table <input type="radio"/> View

Table Selection

Owner:	dbo
Name:	SMOKETEST_ALLTYPES
Column(s):	COLNAMES (Text) KEYFIELD (Text) S_BINARY (Binary) S_CHAR (Text) S_DATETIME (DateTime) S_DECIMALNPFLOAT (Numeric) S_DECIMALNPINT (Numeric) S_DECIMALPNUMERIC (Numeric) S_FLOAT (Float) S_IMAGE (Binary) S_INT (Int) S_MONEY (Currency) S_NUMERICNPFLOAT (Numeric) S_NUMERICNPINT (Numeric) S_NUMERICPNUMERIC (Numeric) S_NVARCHAR (Text) S_REAL (Float) S_SMALLINT (Int) S_TEXT (Text) S_TINYINT (Int) S_VARBINARY (Binary) S_VARCHAR (Text)
Comment:	To be used only after builds

Office

Chapter 4

Creating RealTime Activities

This chapter provides information and instructions for creating RealTime Activities.

Overview

You use the RealTime Activity wizard to create a RealTime Activity. The wizard either prompts you through the steps of creating a RealTime Activity or displays a blank RealTime Activity document, which you can fill in to define the Activity.

Creating a RealTime Activity Using the Wizard

When the User Assistant is turned on, it enables the New RealTime Activity wizard. When you click on the New RealTime Activity icon it starts the wizard, which steps you through the process of creating a RealTime Activity. See the section “How to Create a RealTime Activity Using the Wizard” later in this chapter.

Creating a RealTime Activity without the Wizard

When the User Assistant is turned off, then clicking on the New RealTime Activity icon causes a blank RealTime Activity document to appear. Fill in the fields and select appropriate options in the blank RealTime Activity document. See the section “How to Create a RealTime Activity” later in this chapter.

Usage Requirements

The following usage requirements are necessary for proper RealTime Activity operation:

- **Indexing:** Any database that you are monitoring with a RealTime Activity must be indexed.
- **Key field Datatypes:** Any field used for a key field in a RealTime Activity must be usable as a key field in the back end data source. For each of the data sources listed below, the specified datatypes cannot be used as key fields in RealTime Activities.

Oracle: LONG and LONG RAW types cannot be used as keys.

DB2: BLOB, CLOB, and DBCLOB types cannot be used as keys.

Sybase: TEXT and IMAGE types cannot be used as keys.

ODBC: Varies by specific back end; refer to the database documentation.

Notes: RICH TEXT fields cannot be used as keys.

Refer to the data source documentation for more information about the use of datatypes as key fields.

- **HTTP Server:** When using the HTTP Server to access documents, you should not use RealTime Activity options that rely on the use of hidden fields because the HTTP Server does not save context information.

How to Create a RealTime Activity Using the Wizard

You can use the RealTime Activity wizard to create a RealTime Activity. When the User Assistant is enabled, the wizard steps you through the process of creating the RealTime Activity. If the User Assistant is not enabled, clicking on the RealTime Activity icon causes a blank RealTime Activity document to appear. In that case, see the section “How to Create a RealTime Activity” later in this chapter.

Step 1 – Click on the New RealTime Activity Icon



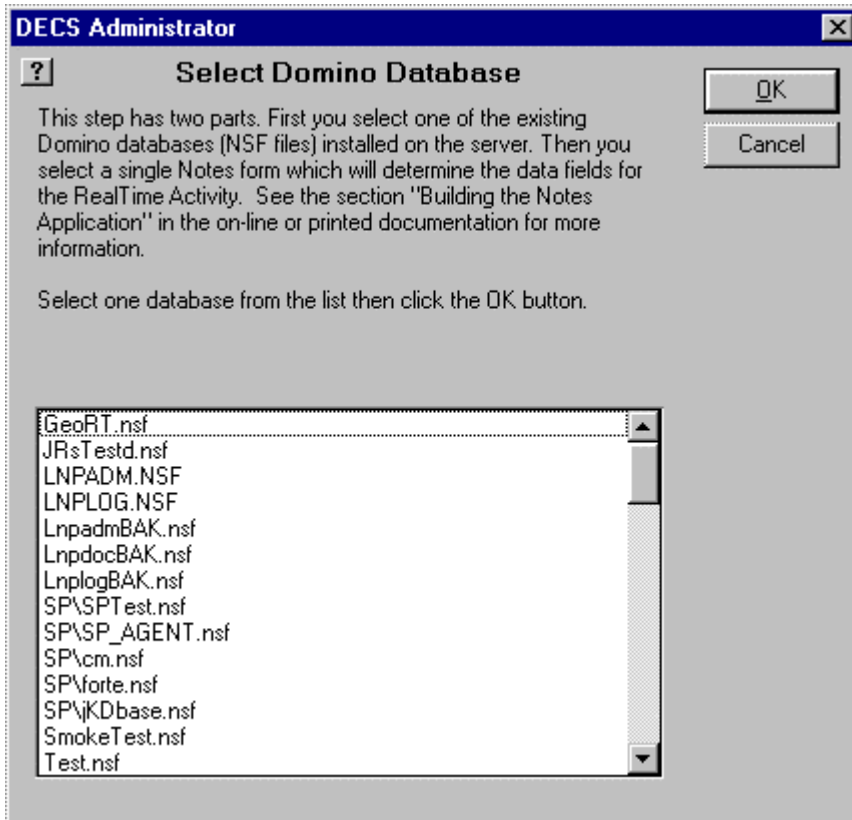
- Click on the RealTime Activity icon to start creating a RealTime Activity.

NOTE: Any choices made while using the wizard can be changed later by opening the RealTime Activity in edit mode.

Clicking the Cancel button returns you to the DECS Administrator if you have not made any selections from the wizard. Once you have made one or more selections through the wizard, then clicking Cancel causes the wizard to shut down and the RealTime Activity document to appear, in edit mode, displaying the choices you have made. You can then manually complete the RealTime Activity document or abandon it.

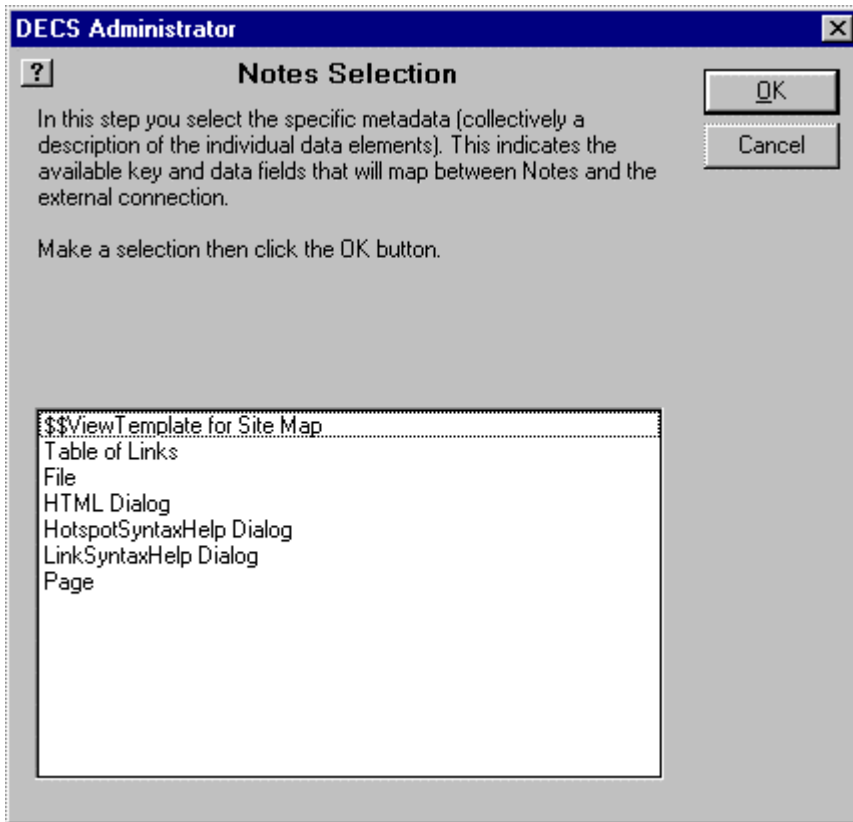
Step 2 – Select the Domino Database to be Monitored

A list of Domino databases appears, as shown below. Select the Domino database that contains the Notes form you want to monitor.



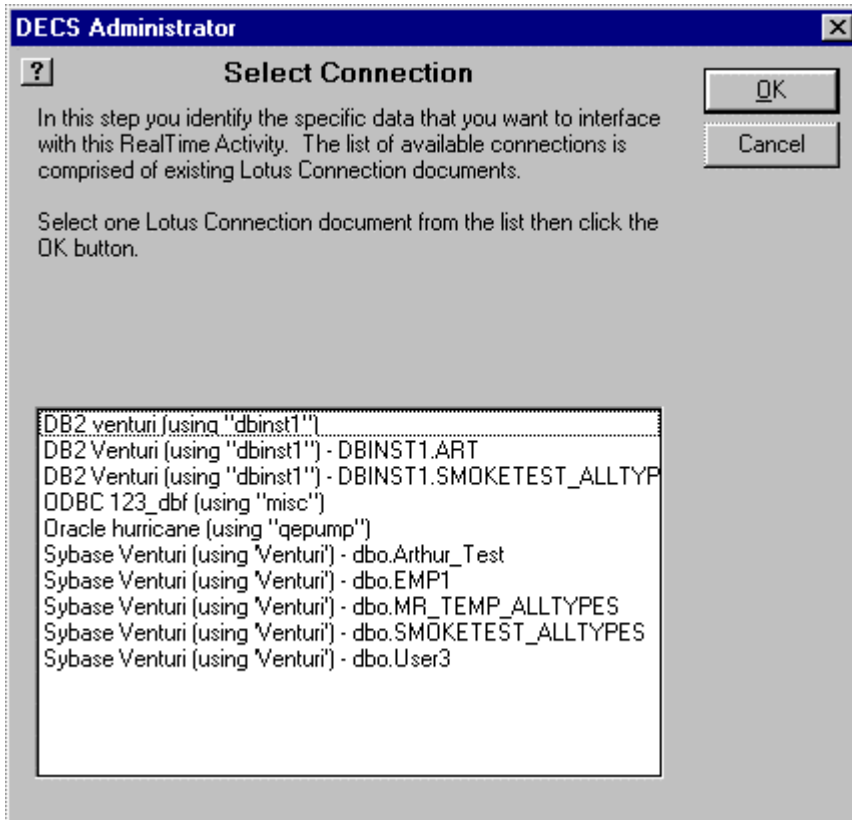
Step 3 – Select the Notes Form to be Monitored

A list of Notes forms within the selected database appears. Select a single Notes form that you want to monitor from the list shown. You can use the Option *Override form* to monitor all forms within a database. See the section on "RealTime Activity Options" later in this chapter.



Step 4 – Select the External Data Source

Select the Connection for external data source you want to monitor from those available in the Select Connection dialog box, shown below.



Step 5 - Map Key Field(s) and Data Field(s)

After specifying the external data source and providing any required connectivity information, such as User Name and Password, the Key and Data Field Mapping dialog box appears. Use this dialog box to map key field(s) and data field(s) between the Notes application and the external data source.

Using the UNID as Key Field

DECS provides an option to use the Notes universal ID that is created automatically by Notes for each document when a document is created. This obviates the need to include an extra field in your Notes front end for use solely as a key field.

DECS Administrator

Key and Data Field Mapping

In this step you configure the mapping of data between the data source and the Notes form that is monitoring the external data source.

Map the fields between the Notes form and the fields in the data source that you are monitoring by clicking on the field names.

You should be familiar with the fields in the Notes form and in the connection before proceeding with this step.

Domino Fields	Keys	External Source Fields
<input checked="" type="checkbox"/> DisplayName (Text) DocumentAccess GroupTitle (Text) GroupType (Text)	DisplayName (Text)	USERNAME (Text)

☐ Use UNID as key

Fields

Domino Fields	Keys	External Source Fields
AvailableForDirSync DocumentAccess <input checked="" type="checkbox"/> GroupTitle (Text) GroupType (Text) ListDescription (Text) ListName (Binary) ListOwner (Binary) LocalAdmin (Binary) Members (Binary) Type (Text)	GroupTitle (Text)	MACHINE (Text)

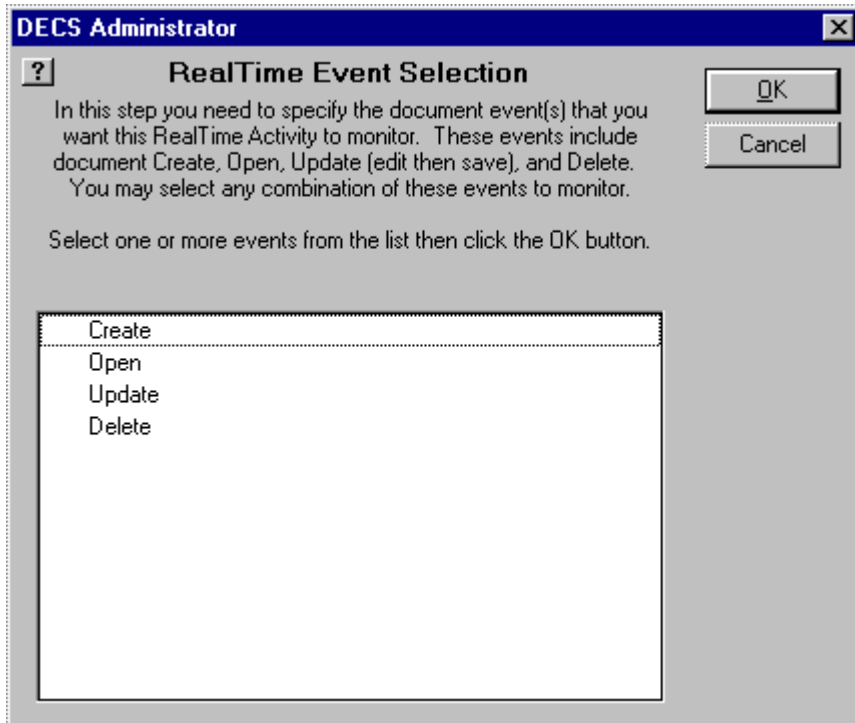
LOGGED_AT (DateTime)
 MACHINE (Text)
☒ USERNAME (Text)

LOGGED_AT (DateTime)
☒ MACHINE (Text)

OK Cancel

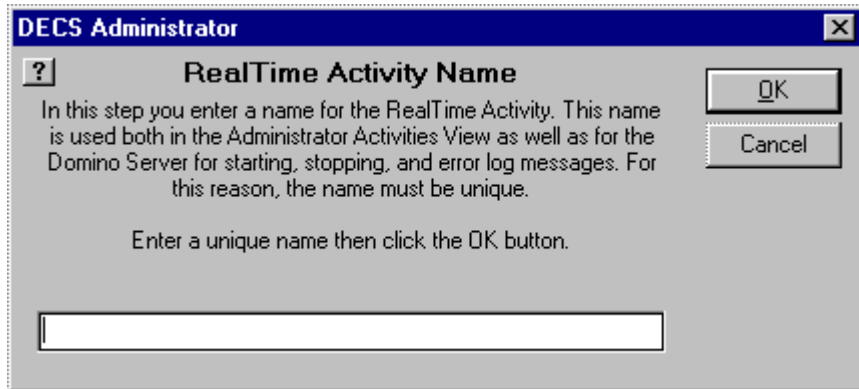
Step 6 – Select the Document Events to Monitor

After mapping key and data field(s), the RealTime Event Selection dialog box appears. The last required step is to select the document event(s) that you want to monitor.



Step 7 – Name the RealTime Activity

Next, you are prompted to enter a unique name for this Activity. Enter a unique name and click OK.



The image shows a Windows-style dialog box titled "DECS Administrator". Inside the dialog, there is a question mark icon in a small box, followed by the title "RealTime Activity Name". Below this, a paragraph of text explains that the name entered will be used in the Administrator Activities View and for Domino Server logs, emphasizing that the name must be unique. To the right of the text are two buttons: "OK" and "Cancel". Below the text, there is a single-line text input field for the user to enter the name. At the bottom of the dialog, there is a final instruction: "Enter a unique name then click the OK button."

DECS Administrator

? **RealTime Activity Name**

In this step you enter a name for the RealTime Activity. This name is used both in the Administrator Activities View as well as for the Domino Server for starting, stopping, and error log messages. For this reason, the name must be unique.

Enter a unique name then click the OK button.

OK Cancel

How to Create a RealTime Activity Using the Wizard

After you name the Activity and click OK, the RealTime Activity document appears, showing your selections. The numbered wizard icons in the document correspond to the major steps the wizard uses to create the RealTime Activity. At this point, you can select the specific options that you want for this RealTime Activity. See the section below, “RealTime Activity Options”, for more information.

RealTime Activity Author: /CAM/Lotus
Current Status: Not Enabled

Identification
Name:

Notes Application
Database: SP Test DB
Name: SP\SPTTestDB.nsf
Form: EmpForm

Lotus Connection
Data Source: Sybase Venturi (using 'Venturi') - dbo.EMP1
Table: dbo.EMP1

Mapping
Key(s): EmpName (Text)
Field(s): EmpNo (Float)

Events
Document Events to monitor:

► **Options ...**

Scheduling
Auto Start: ☐ Enabled


Enter a unique name which will be used to identify this activity at the Domino console and in its log.

How to Create a RealTime Activity

This section provides information on how to create a RealTime Activity without using the wizard. In this case, the User Assistant in the RealTime Activity navigator must be disabled.

When the User Assistant is disabled, clicking on the New RealTime Activity icon causes a blank RealTime Activity document to appear. Follow the steps below to create a RealTime Activity using a blank RealTime Activity document.

Step 1 – Click on the New RealTime Activity icon

Click on the RealTime Activity icon - . The RealTime Activity document appears, as shown below.

How to Create a RealTime Activity

Step 2 – Enter a Name for the Activity

Enter a unique name for the Activity in the Name field at the top of the document.

Step 3 – Select the Notes Application

Click on the down-arrow icon in the Notes Application section. A list of databases appears. Select the database that contains the Notes application that you want to monitor.

Step 4 – Select the External Data Source

Click on the down-arrow icon in the External System section to select from existing Lotus Connection documents. You can create a new Lotus Connection document by clicking on the

NEW icon to the right of the down-arrow button. After creating the new Lotus Connection document, you are returned to the RealTime Activity document from which you clicked the NEW icon.

Step 5 – Map Key and Data Fields

Click on the down-arrow icon in the Mapping section. The field mapping dialog box appears. Use this dialog box to map the key and data fields between the Notes application and the external data source.

Step 6 – Select Event(s) to Monitor

Click on the down-arrow icon in the Events section. The following dialog box appears. Select the document events that you want to monitor. You can select any combination of events.

Step 7 – Select Options

Select the options you want in the Options section of the document. See the section “RealTime Activity Options” later in this chapter for more information.

Step 8 – Save and Close the Document

Select File●Save to save this RealTime Activity definition. Select File●Close to close the RealTime Activity document and return to the RealTime Activity administrator.

Step 9 – Process the RealTime Activity

If you did not schedule the RealTime Activity for automatic starting in the Scheduling section of the RealTime Activity document you can process it using the Connections view and the Start and Stop buttons in the navigator.

Click on the RealTime Activity to select it and then click the Start button to begin processing it.

To stop a RealTime Activity, select the RealTime Activity and click the Stop button in the navigator.

RealTime Activity Options

Depending on the type of event you choose to monitor with a RealTime Activity, additional options are displayed specific to the type of event the activity is monitoring. The figure below shows all of the options categories because the Activity has been configured to monitor each type of event.

RealTime Activity between Test_Sybase2 and Sybase Venturi (using 'Venturi') - dbo.Art_Emp - Lo...
 File Edit View Create Actions Section Window Help
 Edit Document

Author: /CAM/Lotus
 Current Status: Active [view log](#)

Identification
 Name: Art's Sybase test using Art_Emp table

Notes Application
 Database: Test_Sybase2
 Name: Test_Sybase2.nsf
 Form: Test_Sybase2

Lotus Connection
 Data Source: Sybase Venturi (using 'Venturi') - dbo.Art_Emp
 Table: dbo.Art_Emp

Mapping

Key(s):	Field(s):	Key(s):	Field(s):
empno (Float)	empname (Text) job (Text) manager (Text) hire_date (DateTime) salary (Float) dept (Float)	empno (Int)	empname (Text) job (Text) manager (Text) hire_date (DateTime) salary (Currency) dept (Int)

Events
 Document Events to monitor: Create, Open, Update, Delete

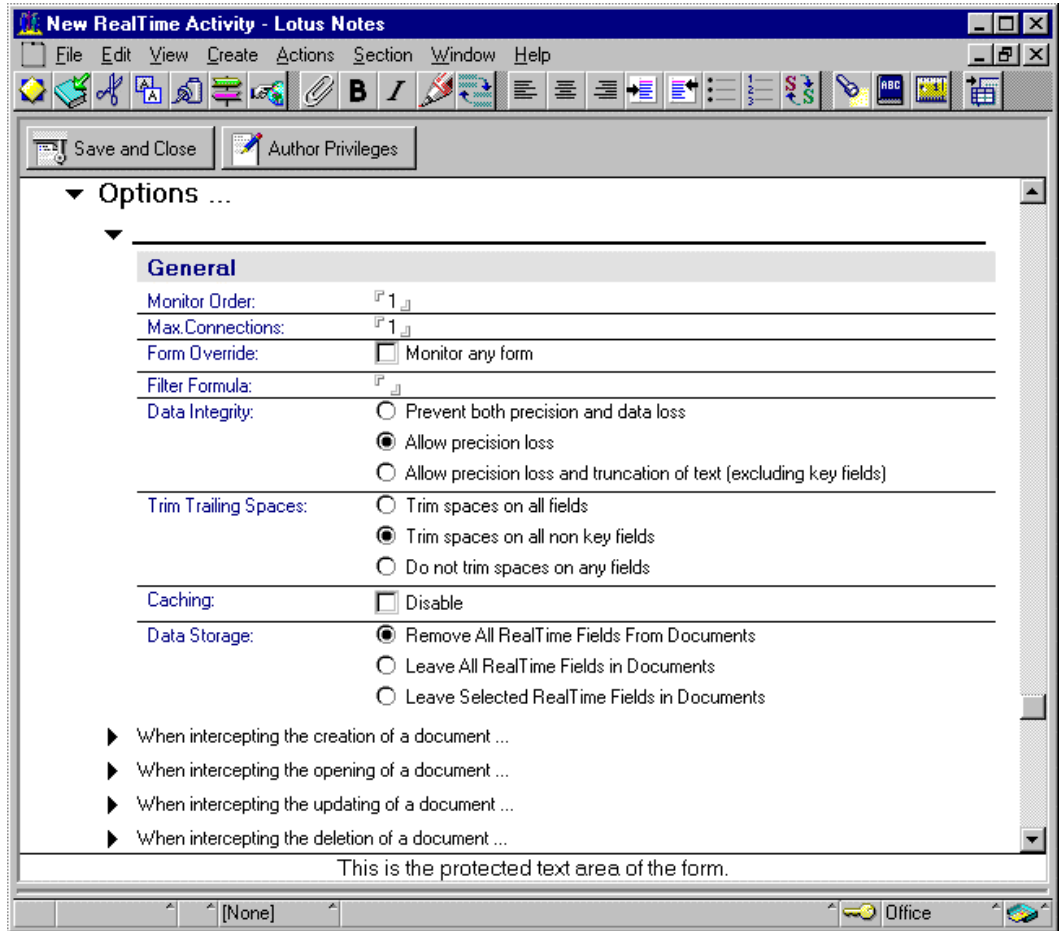
▼ **Options ...**

- ▶ **General**
 - ▶ When intercepting the creation of a document ...
 - ▶ When intercepting the opening of a document ...
 - ▶ When intercepting the updating of a document ...
 - ▶ When intercepting the deletion of a document ...

Scheduling
 Auto Start: ☐ Enabled

General Options

The *General Options* section of the RealTime Activity document, shown below, provides options that can be applied to the RealTime Activity regardless of the type of Event the RealTime Activity is monitoring. Each of the *General Options* is described below.



Option	Description
Monitor Order	<p>If you use more than one RealTime Activity for a single Notes form, you may specify the order in which the RealTime Activities will intercept the document's events.</p> <p>The monitor order also enables you to use multiple RealTime Activities which connect to different tables and use values found by the first RealTime Activity (monitor order 1) as keys for subsequent RealTime Activities (monitor order 2, 3, etc.).</p> <p>NOTE: When adding or updating data to a source from a document that is not the first in the monitor order, a row is created in the source which may not contain all field or key values. To include all data and key fields, use one of the following methods:</p> <ul style="list-style-type: none"> For each RealTime Activity preceding the last RealTime Activity, select the Data Storage option <i>Leave All RealTime Fields in Document</i>. For the last RealTime Activity, indicate to leave specific fields in the document, and list all key fields from the other RealTime Activities. An alternate method is to indicate on each RealTime Activity to leave only those fields in the document that will be referenced by subsequent RealTime Activities.
Max. Connections	<p>This option sets the maximum number of connections to the external database that can be open to service concurrent user requests simultaneously. DECS opens one connection to the external data source when the first Notes application event occurs. If two or more events occur simultaneously, additional connections are made, up to the maximum number of connections specified by this option. When the maximum number of connections is reached, subsequent events are queued and occur when each preceding event is serviced. Each connection lasts only as long as necessary to read from or write data to the external data source. While the connection is persistent, the time required to service each event is minimal, depending on the amount of data being read or written. The maximum number of connections, therefore, does not need to be that great in order to service multiple events. We recommend that you set the maximum connections to 2 or 3, and if users experience significant delays, you can increase this number.</p>
Form Override	<p>The default is to monitor only documents using the form specified in the Notes Application section above.</p> <p>Selecting this option causes the RealTime Activity to process the selected events for all the documents in the Notes database that have the same key(s) as the original metadata, regardless of the form.</p>

RealTime Activity Options

Option	Description	
Filter Formula	An optional Notes formula defining the documents that the RealTime Activity will monitor. Use this option to cause the RealTime Activity to process only documents that satisfy the specified formula.	
Data Integrity	Prevent data loss	Enable this option to have DECS write an error to the log for any data that's lost as a result of the transfer and terminate the transfer.
	Allow precision loss	Enable this option to have DECS not report loss of numerical or datetime precision as a result of the transfer. This allows some loss of precision without stopping the transfer. This option is the default.
	Allow precision loss & truncation of text	Enable this option to have DECS allow precision loss and to truncate text data when necessary to conform to field lengths in the external database. Note that key fields are not truncated.
Trim Trailing Spaces	This option affects any trailing spaces that exist in the Text fields of the external data. Text trimming only occurs when the fields are read from the external source. There are three choices:	
	Trim spaces on all fields	Select this setting to trim trailing spaces from all text fields.
	Trim spaces on all non key fields	Select this setting to trim trailing spaces only from data fields, not from the key fields.
	Do not trim spaces on any fields	Select this option to leave trailing spaces in all fields.
	<p>NOTES:</p> <p>The trailing spaces may be required to ensure matching of fields between Notes and the external data.</p> <p>When data is stored into back end fields of fixed lengths and then retrieved, you may get unexpected results because the back end has padded the data with spaces to fit the fixed length of the field. In such cases, you should use either a variable length field in the back end database, or enable the Trim Trailing Spaces option.</p>	

Option	Description	
Caching	Select this option to disable caching in the HTTP server for monitored documents. When a document is retrieved, the HTTP server in Domino may cache it to avoid disk access for the next retrieval. For occasionally changing backend records, caching may be fine (e.g., an employee handbook); for a RealTime situation with changing data (e.g., checking on an order status), caching should be disabled.	
Data Storage	Remove All RealTime Fields from Documents	Enable this option if you want to remove all the data fields mapped in the Activity from the Notes document before it is saved to disk. This is the default.
	Leave All RealTime Fields in Documents	Enable this option if you want to leave all the data fields in the Notes document, rather than removing them after updating the external source (see above). This option only takes effect when creating or updating a document when the activity is active.
	Leave Selected RealTime Fields in Documents	Enable this option if you want to leave selected data fields in the Notes document. You may want to select this option in order to enable views of the forms. The Static Data button that appears when you enable this option lists the Notes data fields from which you can select. Not selecting any fields is equivalent to enabling the <i>Remove All RealTime Fields from Documents</i> option.
Error Logging	to Log	Logs errors to the Notes Server Log database and to the Activity Log, accessible through the Log button on the Connection Server navigator.
	to Document	Logs errors to Notes document fields. This enables you to take actions based on specific errors or present errors from the external database back to the user. These fields must be included in your Notes form in order for the errors to be displayed.

NOTE: We recommend that you not use the *Allow precision loss & truncation of text* option when storing Rich Text Field BLOB datatypes to binary fields in the back end database. If you do use this option, you should make sure that the back end field is large enough to store the BLOB data.

Document Creation Options

When monitoring Document Creation Events, the following options are available:

Option	Description
Pre-Create Formula	<p>A Notes formula language statement to execute on the new Notes document prior to the creation of a new record in the external database. For example:</p> <pre>FIELD LASTNAME:=@if(LASTNAME = "";"NA";LASTNAME);""</pre>
Stored Procedure	<p>This option executes a stored procedure in the external data source to store data that has been entered in the document. The RealTime key(s) and field(s) are supplied to the stored procedure as input parameters.</p> <p>To see the fields that will be passed to the stored procedure, type the stored procedure name in this field and then press F9. The fields that will be passed to the stored procedure are displayed next to the stored procedure name.</p>

Document Open Options

When monitoring Document Open Events, the following options are available:

Option	Description
Post-Open Formula	<p>Notes formula language statement to execute on the Notes document immediately following the retrieval of the external data. For example:</p> <pre>FIELD FIRSTNAME := @If(FIRSTNAME = "Al"; "Albert" ; FIRSTNAME);""</pre> <p>NOTE: You should not use Post-Open Formulas when using conflict detection.</p>
Stored Procedure	<p>This option executes a stored procedure in the external data source to determine the data that will be retrieved into the document. The RealTime key(s) is supplied to the stored procedure as an input parameter. The stored procedure must produce a result set with both the key(s) and field(s) present.</p> <p>To see the arguments that will be passed to the stored procedure, type the stored procedure name in this field and then press F9. The fields that will be passed to the stored procedure are displayed next to the stored procedure name.</p>
Missing External	If no matching external record is found for a document on open, this

Records	option creates a new record in the external database.
---------	---

Document Update Options

When monitoring Document Update Events, the following options are available:

Option	Description
Pre-Update Formula	Notes formula language statement to execute on the Notes document prior to the update in the external database.
Stored Procedure	<p>When a document is updated you have the option to execute a stored procedure in the external data source to store the data that has been changed in the document. The RealTime key(s) and field(s) are supplied to the stored procedure as input parameters.</p> <p>To see the arguments that will be passed to the stored procedure, type the stored procedure name in this field and then press F9. The fields that will be passed to the stored procedure are displayed next to the stored procedure name.</p>
Conflict Detection	<p>This option ensures that the external data has not changed since the document was opened. If it has changed, the update to the external data source will fail.</p> <p>If you make changes to data in a document and then save the document, you must exit the document before making any more changes if this option is enabled.</p>
Field Level Updates	This option causes the RealTime Activity to not update fields in the external data source unless the corresponding fields in the Notes document have been edited.
Key Field Updates	<p>The following three settings are available for key field updates:</p> <p>Block: Do not allow updates to key fields in the Notes document or the external data source records.</p> <p>Delete/Insert: Updates to key fields will cause the original record in the external data source to be deleted and a new record with the new key added.</p> <p>Ignore: Do not update the key fields in the external data source. If a key field is edited, the change will be stored with the Notes document but the key field(s) in the external data source will not change.</p>

Document Deletion Options

When monitoring Document Deletion Events, the following options are available:

Option	Description
Pre-Delete Formula	<p>Notes formula language statement to execute on the Notes document immediately prior to the deletion in the external data source.</p> <p>Here is an example that sends an email containing information about a deleted document:</p> <pre>FIELD LASTNAME:=LASTNAME; FIELD FIRSTNAME:=FIRSTNAME; @MailSend("John Q Public";"";"";"Doc Deleted"; "Doc deleted from MyDatabase w/ Firstname/Lastname of: ";FIRSTNAME:LASTNAME);""</pre> <p>NOTE: It may be necessary to specify that RealTime data be saved in the Notes document; otherwise, the specified formula may access the field values after they've been deleted. If this were the case for the example above, FIRSTNAME and LASTNAME would be empty.</p>
Stored Procedure	<p>When a document is deleted this option executes a stored procedure in the external data source to remove the data related to the document. The RealTime key(s) is supplied to the stored procedure as an input parameter.</p> <p>To see the arguments that will be passed to the stored procedure, type the stored procedure name in this field and then press F9. The fields that will be passed to the stored procedure are displayed next to the stored procedure name.</p>

Logging of RealTime Activity Status

Information about DECS operations is logged to the Domino Server log in the Miscellaneous section of the log. It provides the time and date when a RealTime Activity started and ran, and information about any errors that occurred during processing of the RealTime Activity.

Note that if you select the General Option *Log Errors to Document*, then any error information is included in the RealTime Activity document.

Chapter 5

Building the Notes Application

This chapter provides information about building a Notes application for use with a RealTime Activity.

Building the Notes Application for a RealTime Activity

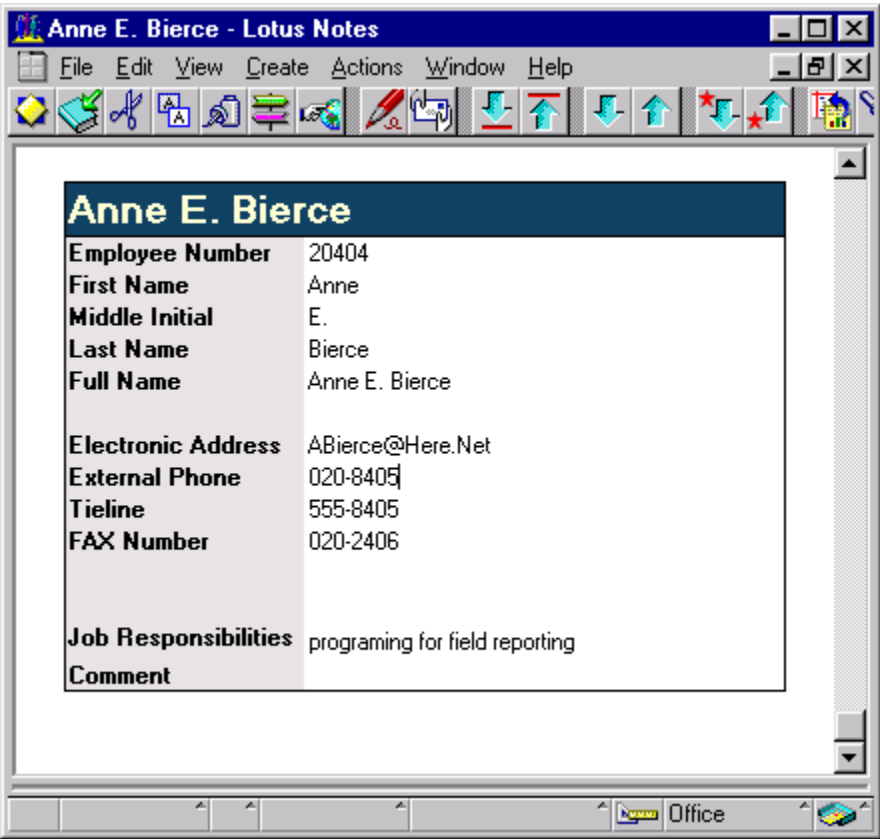
The Notes application must include at least one field that maps to a key field in the external data source. A key field is a field or fields used to uniquely identify the data. The RealTime Activity can include multiple key fields.

The Notes application being monitored by the RealTime Activity must also include data fields that map to data fields in the external data source.

The next section gives an example of an application that could be used with a RealTime Activity.

Example Application

Shown below is a Notes application that monitors employee information in an external data source.



Example Application Design View

Shown below is the design view of the example application. This application provides labels next to the data fields that map to the data fields in the external data source this application is monitoring. The RealTime Activity that uses this application also defines the field mapping.

Example Application

Person - Form - Lotus Notes

File Edit View Create Design Text Table Window Help

FullNameDisp

Employee Number

EmpID

First Name

FirstName

Middle Initial

MiddleInitial

Last Name

LastName

Full Name

FullName

Electronic Address

MailAddress

External Phone

ExternalPhoneNumber

Tieline

TielinePhoneNumber

FAX Number

FAXPhoneNumber

Public Key

PublicKey

Certified Public Key

Certificate

Client License Type

ClientType

Job Responsibilities

JobResponsibilities

Comment

Comment

Type

Define: Person (Form)

Event: Window Title

Run: ☐ Simple action(s) ☒ Formula ☐ Script ☐ Show browser

@V2If(@IsNewDoc; "New Person"; @Implode(@Word(@Explode(@Subset(FullName; 1); "/"); "="; -1); "/"))

Fields & Functions...

@Commands...

Helv 9 [None] Office

Example RealTime Activity

Shown below is an example of a RealTime Activity that uses the example application. The field mapping in the Mapping section of the document maps to the example application above.

RealTime Connection between Sample Phone Book and Notes Breeze demo\phonedat.nsf - Per...

File Edit View Create Actions Link Window Help

RealTime Activity Author: Scott Prager/CAM/Lotus
Current Status: Not Enabled Log: [icon]

Notes Application		External System	
Database:	Sample Phone Book	Data Source:	Notes Breeze demo\phonedat.nsf - Person
Name:	demo\phone.nsf		
Form:	Person	Form	Person

Mapping	
Key(s):	LastName (Binary) FirstName (Binary)
Field(s):	Certificate (Text) ClientType (Text) Comment (Binary) EmplID (Float) ExternalPhoneNumber (Binary) FAXPhoneNumber (Binary) FullName (Binary) JobResponsibilities (Binary) MailAddress (Text) MiddleInitial (Binary) PublicKey (Text) TelinePhoneNumber (Binary) Type (Text)

Events	
Document Events to monitor:	Open, Update

► Options ...

Chapter 6

Examples using RealTime Activity Options

This chapter provides examples of RealTime Activities that utilize the options available in the RealTime Activity document.

Using Filter Formulas

This section provides examples of how you would use the Filter Formula option in conjunction with the Monitor Order option to achieve specific results.

Saving Data to an External Data System

Scenario: You want to store data from a Notes document in an external data system, such as a DB2 or Oracle database, when the document data becomes stable.

When each document is stable and ready to store on the external system, you would change its status from “under review” to “publish.”

How to Configure: In this case, you would set up a RealTime Activity that monitors the Notes application.

In the RealTime Activity, you would use a filter formula that includes only documents that have been marked “Published.”

At that point the information from the Notes document will be written to the DB2 database, where it would be available to other applications. Any documents that are still “under review” will not be stored in DB2.

Accessing Different External Sources using the Same Notes Application

Scenario: You want to use a single Notes front end to interact with, but potentially the North East sales information is in a different table than the South West region.

In this case, you would set up one RealTime Activity for each region with different external data sources, but using one Notes document as the front end.

You could use a filter formula that would detect the region of information/account request and go to the appropriate back end data.

Scenario: Perhaps you have two product catalogs in different databases but you want to present the data through the same front end Notes application. You want to use a single Notes front end, but dependent on a field, you want to control where the data is coming from.

Set up identical RealTime Activities monitoring a single Notes application, but have each RealTime Activity use a filter formula to handle part of the request.

Using Data Storage Options

This section provides some examples of how and when you would use the Data Storage options in RealTime Activities.

Building Views

Any data that you want to use when building views must reside in the Notes document. Use the option *Leave Selected RealTime Fields in Document*. Select the fields that you want to reference in a view from those fields available when this option is selected.

Storing a Static Copy of External Data

If you want to store a static copy of the external data, you would select *Leave All RealTime Fields in Document*. Note that if the back end data changes, the data in the document is not updated until the document is reopened.

Using Monitor Orders

Scenario: The external data for your Notes Human Resources Information system is in multiple tables. A "join" of the data is required to tie these various tables together to populate the single Employee Information form in the Notes application. In some cases, a join from the database may not be possible, as the tables may reside in different databases (one in Oracle, one in Sybase, for example). Information in the Employee Information form includes Employee Name, Department Name and Location Information. The following table explains which fields are used from which table and how they will relate to the Notes form.

Fields Used:

Employee Information Form	EMPLOYEE Table	DEPARTMENT Table (access with monitor order 2)	LOCATION Table (access with monitor order 2)
EmpID	EmployeeID		
EmpName	EmployeeName		
DepartmentNo (may be hidden) Department	DeptNo	DeptNo DeptName	
DepartmentLoc (may be hidden) Location		DeptLoc	LocCode LocName ...

How to Configure: To do this join, you will create three RealTime Activities that each monitor separate tables: one to monitor the Employee table, one to monitor the Department Table and one to monitor the Location Table. The first Activity (using a Monitor Order of 1) will provide data which will be used by the other Activities (with Monitor Orders of 2 and 3) as "keys" to their tables. The Notes Form "Employee Information" will need to contain those fields that will act as "keys" to the secondary tables. These fields may be "hidden" from the user. If updates will be made through the Notes application to these secondary tables, select the Option "Leave selected realtime fields in document" and list all the fields which are used as keys to the backend (DeptmentNo, DepartmentLoc).

1. Create a connection document for each of the external data sources: one for the employee table, one for the department table, one for the location table.

Using Monitor Orders

2. Create a RealTime Activity based on the Employee Information Form and the Employee Table. Select all fields that will be used by the form and/or used as a key to another field. In this case, select at least the following: EmployeeName, DepartmentNo.

Notes	Employee Table
EmplID as the key	EmployeeID as the key
DepartmentNo	DeptNo

3. Create a second RealTime Activity based on the Employee Information Form and the Department Table. Select the following fields:

Notes	Department Table
DepartmentNo as the key	Deptno as the key
Department	DeptName

Set the Monitor Order for this activity to 2. This will allow the first activity to retrieve the information from the employee table before looking into the department table for the DeptName field.

4. The third activity will work like the second, except it monitors the Location Table. It will use a monitor order of 3 because its lookup is dependent on information provided by the second Activity.

Notes	Location Table
DepartmentLoc as the key	LocCode as the key
Location	LocName

Using Stored Procedures

This section provides an example of a RealTime Activity that uses stored procedures with a Sybase external data source.

In this example there are four stored procedures for the table **addrbook** in Sybase. When using these stored procedures, the key fields used must be FirstName then LastName and the mapped fields must be (MailDomain, MailServer, MailAddress, CompanyName, and State). The stored procedures, table name and fields must use the correct case since Sybase is case sensitive.

To use a stored procedure, enter the name of the stored procedure in the options section for the appropriate event in the RealTime Activity document. In this example, these would be:

Create: **QEInsert**addrbook

Open: **QESelect**addrbook

Update: **QEUpdate**addrbook

Delete: **QeDelete**addrbook

Using Stored Procedures

Below is an example of a RealTime Activity document showing how these stored procedures are entered in the Stored Procedure fields of the corresponding document event options.

New RealTime Activity - Lotus Notes

File Edit View Create Actions Text Window Help

Save and Close Author Privileges

Events

Document Events to monitor: ☐ Create, ☐ Open, ☐ Update, ☐ Delete

▼ **Options ...**

► **General**

▼

Document Create Event

Pre-Create Formula: ☐

Stored Procedure: ☐ QEInsertaddrbook { MailDomain, MailServer, MailAddress, CompanyName, State, FirstName, LastName }

▼

Document Open Event

Post-Open Formula: ☐

Stored Procedure: ☐ QESelectaddrbook { FirstName, LastName }

Missing External Records: ☐ Create Record ☒ Generate Error ☐ Ignore

▼

Document Update Event

Pre-Update Formula: ☐

Stored Procedure: ☐ QEUpdateaddrbook { MailDomain, MailServer, MailAddress, CompanyName, State, FirstName, LastName }

Conflict Detection: ☒ Check External Data for Changes*

Field Level Updates: ☐ Only Update Changed Fields*

Key Field Updates: ☐ Block* ☐ Delete/Insert * ☒ Ignore

* These options create temporary hidden fields in opened documents

▼

Document Delete Event

Pre-Delete Formula: ☐

Stored Procedure: ☐ QEDeleteaddrbook { FirstName, LastName }

Key field update options.

Office

Using the Stored Procedures in SQL

The examples below show how you would execute the stored procedures in SQL through the database client, in this case Sybase. You can test that the stored procedures work properly by executing them directly as shown below.

```
execute QESelectaddrbook 'FirstName','LastName'
```

```
execute QEInsertaddrbook  
'Adelino','Fontes','MailDomain','MailServer','MailAddress','CompanyName','State'
```

```
execute QEUpdateaddrbook 'Adelino','Fontes','Edge','Tempest','Ports','Edge Research','NH'
```

```
execute QeDeleteaddrbook 'Adelino','Fontes'
```

Stored Procedure Definitions

Below are the actual stored procedure definitions used in the preceding example. Note that these are Sybase procedures; stored procedures for other data sources may be different.

```
create procedure dbo.QESelectaddrbook(
```

```
@FirstName varchar(20),
```

```
@LastName varchar(20)
```

```
)
```

```
as
```

```
select FirstName,LastName,
```

```
MailDomain, MailServer, MailAddress,
```

```
CompanyName,State from addrbook
```

```
WHERE FirstName = @FirstName and
```

```
LastName = @LastName
```

```
create procedure dbo.QEUpdateaddrbook(
```

Using Stored Procedures

```
@FirstName varchar(20),  
  
@LastName varchar(20),  
  
@MailDomain varchar(20),  
  
@MailServer varchar(20),  
  
@MailAddress varchar(20),  
  
@CompanyName varchar(20),  
  
@State varchar(20)  
  
)  
  
as  
  
UPDATE addrbook SET  
  
MailDomain=@MailDomain, MailServer=@MailServer,  
  
MailAddress=@MailAddress,  
  
CompanyName=@CompanyName, State=@State  
  
WHERE  
  
FirstName=@FirstName and LastName=@LastName
```

```
create procedure dbo.QECreateaddrbook(  
  

```

```
@FirstName varchar(20),  
  
@LastName varchar(20),  
  
@MailDomain varchar(20),  
  
@MailServer varchar(20),  
  
@MailAddress varchar(20),  
  
@CompanyName varchar(20),  
  

```



```

@State varchar(20)

)

as

INSERT INTO addrbook

(FirstName,LastName,

MailDomain, MailServer,

MailAddress, CompanyName,State)

VALUES

(@FirstName, @LastName,

@MailDomain, @MailServer,

@MailAddress, @CompanyName, @State)

create procedure dbo.QEDeleteaddrbook(

@FirstName varchar(20),

@LastName varchar(20)

)

as

DELETE FROM addrbook

WHERE FirstName=@FirstName and LastName=@LastName

```

Chapter 7

RealTime Dynamic Queries

This chapter provides information on how to use DECS to create RealTime Activities that enable dynamic, on-demand queries to external data sources.

Overview

The DECS RealTime Activity enables a Domino database form to provide direct access to back end data sources supported by DECS. The RealTime Activity form requires that one or more key field values be held in common with the Notes form (accessed by the Notes client or web browser client) and the back end source. It is the key field(s), entered into the Notes form, that controls the query of the back end source data.

Normally, the RealTime Activity requires that the key field(s) exist in the Notes stub document as well as the back end data source. In this configuration, the Notes or web client would open the form that has the appropriate key field value you want to use to query additional back end data. There are, however, many situations where the key field value will not already be available within a Notes document. In such cases, clients want to freely insert the key value when they open the Notes forms.

The RealTime Activity can be set to allow web and Notes client end users to open Notes application forms and enter key values to Notes form fields then dynamically trigger DECS. These entries are then sent via the Domino DECS server to query the back end data source. As an example, a customer with a package tracking number opens a Notes form using a web browser, enters the tracking number to the Domino server web form, then, using a button on the form, submits the new document to the Domino server. The button causes the document to be saved and then reopen the new document. RealTime Activity detects the open event and uses the tracking number to query the source database, which locates the package information using the tracking number as a key. The matching record results are inserted into the Notes document and sent back to the client web browser, which displays the status of the package, in real time.

The next sections provide information required for defining the RealTime Activity and preparing Notes application forms to accept dynamic data queries from web clients and Notes clients.

RealTime Dynamic Queries from Web Clients

This section provides information about using your Web client to invoke a RealTime Activity that accepts a key value input for querying an external data source. This extends the functionality of the RealTime Activity to provide a dynamic query capability.

Overview of Steps

1. Create a Notes application form that defines the format of the data to be returned to end-users when the query is processed. (This is what is to be displayed to the user accessing the Domino server via a web client.) This form includes the fields that you want populated from the back end database. Also include one or more key fields, in which the user will enter values to be used to locate the correct external data. The appearance of the application form may be improved by using the Notes form design option of the "Hide paragraph when..." property box. In the example, the form is broken into three primary sections. The top of the form is always displayed. The input field, input help, and "Locate" button only display when the document is in edit mode. The data fields only display when the document is in read mode.

In our example, the key field is "PackageID". Also, include one hidden field in the form:

`$$Return` field - A computed for display only field of type "text" that specifies the URL for the new document returned once the query data has been input and "Locate" button has been selected.

2. Within the overall web application, create a URL link to the application form (for more information on specifying URLs to Notes databases and forms, see the Notes Application Developer's Guide). In our example, we provide an "About" page for the database that contains the URL link for the application form.
3. Define a DECS RealTime Activity that monitors the Notes database and the form defined in step 1 above. This form should include the key and data fields you define in step 1 above.
4. Access the Domino Lookup form from a Web browser. Type in a tracking number (PackageID field). When the Locate button is selected, the document is saved in the database and the `$$Return` text is passed to the Domino HTTP process. This results in the new document being re-opened. The RealTime Notes Connection is activated, taking the key value, from the PackageID field, submitted in the form. The PackageID value is then sent to the back end source, and the table records are searched according to the PackageID value. Results of the search are sent back through DECS and inserted into the document. The document is then send from the Domino server to the Web client as directed by the

URL specified in the \$\$Return field.

Using the "Hide paragraph when..." properties, different sections of the form appear under different conditions.

Section 1 is always displayed.

Section 2, including the input field, the button, and the help text, only appears when the document is in edit mode.

Section 3 is always hidden. The Domino server only references it.

Section 4 is error text. It is displayed when the document is in read mode and, referencing the "status" field, when no data has been inserted into the document from the RealTime Activity.

Section 5 contains the data inserted by DECS and only displays in read mode and, once again referencing the "status" field, when data exists.

Courier Services, Inc.

Locate

Stuff

Stuff

Stuff

Stuff

1

Tracking

Track any Courier bar coded shipment - any time, any where in the world, instantly.

Notice

Courier authorizes you to use Courier tracking systems solely to track shipments tendered by or for you to Courier for delivery and for no other purpose. Any other use of Courier tracking systems and information is strictly prohibited.

2

Tracking number:

PackageID

Locate

Use the format example below to help you enter your tracking number correctly.

999-999-9999

3

Return

4

The system is not able to process your request at this time. Please try again later. We apologize for any inconvenience this may have caused.

5

Current status:

Status

Delivered on:

DateReceived

Received by:

Recipient

Sent on:

DateSent

Addressed to:

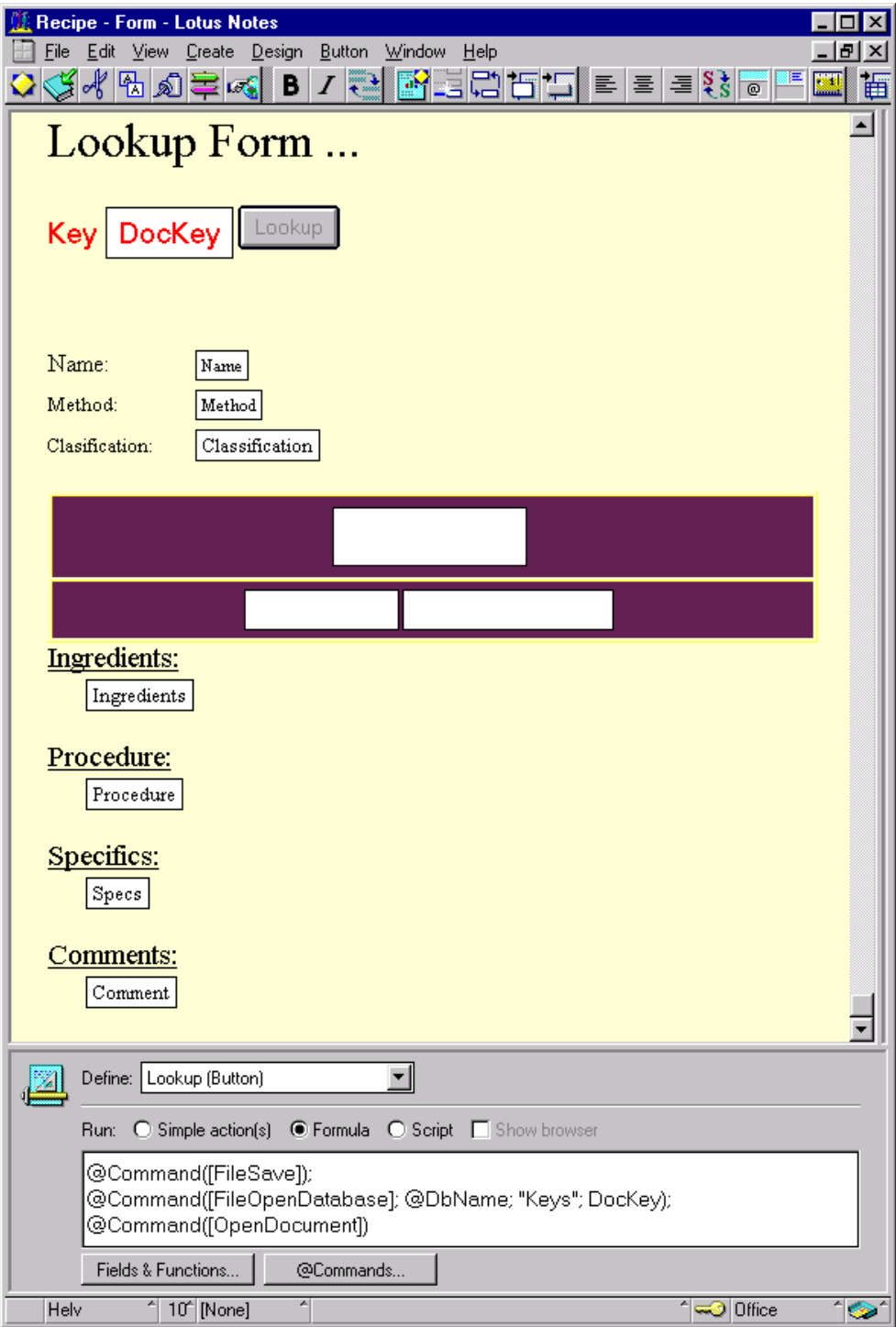
Address

Courier service:

ServiceType

RealTime Dynamic Queries from Notes Clients

1. Edit your Notes database.
2. If one does not exist, add a view sorted on the RealTime key or a column formula if you use more than one key field. This key or formula value will be used in place of the ***bold italicized text***. The example uses a single key field called ***DocKey***.
3. The view is referenced by the button below in place of the **bold text**. The example uses a sorted view called **Keys**.
4. Edit your Notes form.
5. Identify the RealTime key(s) field.
6. Add a button to the Notes form.
7. Add the following formula to the button, substituting the above view name for the red text, and the RealTime key (or formula text if using more than one key) for the blue text. Note that the blue text must match the key or column formula of the view.
 - @Command([FileSave]);
 - @Command([FileOpenDatabase]; @DbName; "**Keys**"; ***DocKey***);
 - @Command([OpenDocument])
8. Optionally use hide formulas to make the button and key field(s) display when composing a new document and all fields display when opening an existing document.
9. Save the form.
10. With RealTime running, compose a document. Clicking the button will save the document with the new key and re-open it, displaying the RealTime loaded data. This has an added side effect of allowing the user to jump back to compose the form and change the key. Pressing the button locates a different document.



Appendix A

Configuration and Troubleshooting

This appendix provides information about INI variables, error messages and known problems that you may encounter under certain conditions while using DECS.

NOTES.INI Variables

The following entries may be added to the NOTES.INI file to control aspects of DECS:

DECSTranslation: This controls text translation, allowing the user to increase performance in exchange for certain assumptions about the data being accessed. Note that none of these settings affects translation between unicode and other character sets, since it is always required. There are three valid numeric settings:

- 0 Do not perform translation between character sets (except Unicode). This is valid when all data being accessed is compatible with the Notes LMBCS character set - primarily ASCII printable characters.
- 1 No not performed translation between non-LMBCS (and non-Unicode) character sets. This is valid when all data being accessed, except for Notes LMBCS data, is in compatible character sets.
- 2 Always translate between any character sets. This is the default.

DECSNativeText: This allows the local machine's native character set to be overridden. There are various situations when the native character set is used within DECS, and some backend systems always consider client data to be in the native character set. Setting this value to a valid text format string replaces the character set obtained from the operating system by DECS with the indicated character set. Appendix D of the LSX LC documentation (LSXLCDOC.NSF) lists supported character sets. Use the text that remains after removing the "LCSTREAMFMT_" prefix. For example, Code Page 932, represented by the constant LCSTREAMFMT_IBMCP932, would be set as follows:

DECSNativeText=IBMCP932.

Installing NotesPump 2.5a after DECS

If you install Lotus NotesPump 2.5a after installing DECS on your Domino Server, you may encounter errors related to opening documents.

To fix this, edit the following line in the NOTES.INI file:

```
EXTMGR_ADDINS =
```

To read as follows:

```
EXTMGR_ADDINS = decsext.dll, lnpxt.dll
```

This setting allows both the DECS RealTime Activities and the NotesPump Activities to function properly from the same Domino server.

Using DECS with Notes Clients Older than Release 4.62

If Notes clients older than Release 4.62 will be accessing DECS to create and run RealTime Activities, you should add the following setting to your NOTES.INI file on your Domino Server in order to enable support for proper error messages for these older clients:

```
DECSOldClientSupport = 1
```

Note that this setting causes DECS to substitute older versions of error messages to *all* clients. This setting is not necessary for installations where all clients are Release 4.62 clients.

Using DECS on Solaris Platforms

To use DECS, Solaris users must remove the setuid bit from the Domino Server executable. Additionally, to ensure proper performance of the Domino Server, the system configuration file `/etc/system` needs to be updated to allow a larger than default amount of file descriptors per process. Please follow the steps below as the root user only after verifying that you have installed all the required Solaris OS patches as described in the Domino Release Notes:

1. Remove the setuid bit:

```
(Solaris Sparc)      chmod u-s /opt/lotus/notes/latest/sunspa/server
```

(Solaris Intel Edition) `chmod u-s /opt/lotus/notes/latest/sunx86/server`

2. Backup the system configuration file:

```
cp /etc/system /etc/system.bck
```

3. Update the system configuration file:

Method A: Using an editor such as `vi`, edit `/etc/system` and add the following line:

```
set rlim_fd_max=8192
```

Method B: execute the following command that will update the file:

```
echo set rlim_fd_max=8192 >> /etc/system
```

4. Reboot the system for the configuration change to take effect.

Failure to remove the setuid bit will result in the Domino Server emitting the following message when using DECS:

```
"Addin: Agent error message: Error loading USE or USELSX module: *lsxlc"
```

Error Messages

Cannot use field ['FIELDNAME'] as both a key and a data field

Fields provided for RealTime activities must be either key fields or data fields - a field cannot be used as both in a single RealTime activity (although one field can be used as a key in one activity, and data in another activity). Remove any field provided as both a key field and a data field from at least one of those lists.

Failure accessing shared RealTime Activities table

An internal error was encountered when attempting to access activity information. Record as much information as possible about the circumstances and contact Lotus technical support.

Failure encountered in monitoring process -- ERROR MESSAGE

Errors which are generated by the realtime monitors in the Domino server are logged with this prefix. The complete error text from the realtime error is appended.

Unexpected internal failure in RealTime monitoring

An internal error was encountered when attempting to access context information. Record as much information as possible about the circumstances and contact Lotus technical support.

Update of key field ['FIELDNAME'] is not permitted

Key values in a document were altered, but the realtime activity indicated that changes to key fields should be blocked. Updates to both Notes and the backend system were aborted.

This record has changed in the backend database since being opened - action cancelled

The realtime activity option to check the external system for changes before writing changes to the backend was enabled, and the check indicated changes in the backend. Since the document was opened, another system or client changed the corresponding external record.

NOTE: This error can be erroneously generated in Domino 4.x systems when using the Notes client UI to save a document without closing it, and then saving it again. To avoid this problem on monitored documents with integrity checking enabled, close a document after saving changes and reopen it.

Cannot locate corresponding external record

The key values in the opened document being monitored by a realtime activity did not correspond to a record in the external system. This error can be suppressed by selecting the realtime activity option to create the external record if it doesn't exist - instead of the error, a new record corresponding to the current Notes document data will be created.

Failure compiling Filter Formula: FORMULA COMPILATION ERROR

The filter formula provided for the realtime activity failed compilation. The compilation error is generated with the error message. Fix or remove the filter formula to successfully run the realtime activity.

Failure compiling Pre-Open Formula: FORMULA COMPILATION ERROR

The pre-open formula provided for the realtime activity failed compilation. The compilation error is generated with the error message. Fix or remove the pre-open formula to successfully run the realtime activity.

Failure compiling Post-Update Formula: FORMULA COMPILATION ERROR

The post-update formula provided for the realtime activity failed compilation. The compilation error is generated with the error message. Fix or remove the post-update formula to successfully run the realtime activity.

Failure compiling Post-Create Formula: FORMULA COMPILATION ERROR

The post-create formula provided for the realtime activity failed compilation. The compilation error is generated with the error message. Fix or remove the post-create formula to successfully run the realtime activity.

Failure compiling Post-Delete Formula: FORMULA COMPILATION ERROR

The post-delete formula provided for the realtime activity failed compilation. The compilation error is generated with the error message. Fix or remove the post-delete formula to successfully run the realtime activity.

Error Messages

"Unknown OS error: libdecsext.*"

This is a Notes error reporting that Notes couldn't load the DECS extension manager library. Check that you have properly installed and configured DECS. If problems persist, contact technical support.

"DECS Server addin task initialization failed"

This indicates that the DECS addin task startup encountered an error. Check that you have properly installed and configured DECS. If problems persist, contact technical support.

"DECS Server is unable to allocate addin task resources"

The DECS Server is unable to allocate additional resources. Check that you have properly installed and configured DECS. If problems persist, contact technical support.

"DECS Server cannot connect to external system"

You may not have the proper connectivity software installed that is required for accessing the external data system. Refer to the *Domino Connectors Setup Guide* for information about the native software required for connectivity to each of the DECS supported data sources.

"DECS Server cannot find external table/metadata"

The metadata selected for this activity does not exist in the back end data source.

"DECS Server cannot find external procedure/transaction"

The document open event captured by DECS encountered an error. The error details are logged to the Domino Server log.

"DECS Server error retrieving external record"

The document open event captured by DECS encountered an error. The error details are logged to the Domino Server log.

"DECS Server error inserting external record"

The document creation event captured by DECS encountered an error. The error details are logged to the Domino Server log.

"DECS Server error updating external record"

The document update event captured by DECS encountered an error. The error details are logged to the Domino Server log.

"DECS Server error deleting external record"

The document deletion event captured by DECS encountered an error. The error details are logged to the Domino Server log.

"DECS Server cannot locate the corresponding record in the external system"

The document key field values do not correspond to a record in the back end data source. The record in the external data system may have been deleted.

"DECS Server unable to update document due to key field changes; changes to key fields have been disabled"

The key fields in the external data have been modified since the document has been opened. To allow key field changes, select the appropriate setting for the option *Key Field Updates* in the Document Update options section of the Activity document.

"DECS Server unable to update document due to conflict; the external record has been modified since being opened"

The external data has been modified by another application since the document has been opened. Close and re-open the document.

"DECS Server data overflow accessing external record"

For document open events, this indicates that data in an external field was too long. Usually this is due to text longer than 64K. To avoid this problem, change the Notes field to Rich Text. (See the error in the Domino Server log for information on which field caused the overflow.)

For document updates and inserts, this message indicates that the document data overflowed a back end field. You may need to change the data type in the back end to store large amounts of data.



Part 3:

LotusScript

Extension for

Domino

Connectors

Reference Guide

Chapter 1

Introduction

This chapter provides an introduction to the LotusScript Extension for Lotus Domino Connectors and gives information on the organization of this manual.

This chapter also gives a glossary of common terms and concepts used throughout the documentation, a general description of the individual classes of the LotusScript Extension for Lotus Domino Connectors, and describes how the classes might be used to implement a typical application. The documentation assumes a working knowledge of LotusScript, the Notes development environment, and the Notes classes. For more information on any of these subjects, please refer to the *Notes Programmer's Guide*.

Organization of this manual

The table below describes the organization of this documentation and the information contained in each section.

Chapter	Description
Chapter 1 Introduction	This chapter provides an introduction to the LotusScript Extension for Lotus Domino Connectors. It also includes information about the organization of this manual.
Chapter 2 LCConnection Class	This chapter provides descriptions of the LCConnection Class methods and properties, and includes examples for using each of the methods.
Chapter 3 LCCurrency Class	This chapter provides descriptions of the LCCurrency Class methods and properties, and includes examples for using each of the methods.
Chapter 4 LCDatetime	This chapter provides descriptions of the LCDatetime Class methods and properties, and includes examples for using each of the methods.
Chapter 5 LCField Class	This chapter provides descriptions of the LCField Class methods and properties, and includes examples for using each of the methods.

Organization of this manual

Chapter	Description
Chapter 6 LCFieldlist Class	This chapter provides descriptions of the LCFieldlist Class methods and properties, and includes examples for using each of the methods.
Chapter 7 LCNumeric Class	This chapter provides descriptions of the LCNumeric Class methods and properties, and includes examples for using each of the methods.
Chapter 8 LCSession Class	This chapter provides descriptions of the LCSession Class methods and properties, and includes examples for using each of the methods.
Chapter 9 LCStream Class	This chapter provides descriptions of the LCStream Class methods and properties, and includes examples for using each of the methods.
Appendix A Error Messages	This appendix provides a list of error messages that can occur during a script execution, and provides a description of the error message format.
Appendix B Property Tokens	This appendix provides a list of property tokens that are used in some of the LotusScript Extension for Lotus Connectors methods.
Appendix C Connector Properties	This appendix provides a list of the properties for each Lotus Connector.
Appendix D Character Sets	This appendix provides a list of character sets supported for use with the LotusScript Extension for Lotus Connectors.

Overview of the LotusScript Extension for Domino Connectors

Lotus Domino Connectors provide native access to a wide variety of DBMS products, ODBC, the platform File system, Enterprise Resource Planning systems, and Transaction Processing systems. The LotusScript Extension for Lotus Domino Connectors (LSX LC) extends these Connectors to LotusScript.

LotusScript provides an integral programming interface to Lotus Notes. The LotusScript Extension for Lotus Domino Connectors enhances the power of Notes by extending its scripting to data outside of Notes. The programming model is independent of the individual Connector. This eliminates the need to learn each individual system, while at the same time allowing experienced users to access the individual features of a specific system.

For example, through Lotus Connectors, Notes and web applications have the ability to retrieve and act upon data within agents, during document events, or at the click of a button.

This release of the LSX LC supports access to the following Connectors:

- DB2/UDB
- EDA/SQL
- File System
- Notes
- ODBC
- Oracle
- Sybase

It is important to note that the LotusScript Extension for Lotus Domino Connectors may be used alone or in conjunction with the Domino Enterprise Connection Services (DECS). Respectively, these two technologies provide programmatic and declarative access to external data for application development.

Connectivity Software Requirements

Access to supported Lotus Connectors may require software to be installed on the Domino Server or the Notes client from which the Lotus Connector scripts are run. Refer to the *Domino Connectors Setup Guide* for information about software that may be required in order to access any particular data source. The *Domino Connectors Setup Guide* is provided online in NSF format (LCCON.NSF).

Registration and Loading of the LSX LC

The LSX LC is registered when the Domino Server is installed.

You must load the LotusScript Extensions for Lotus Connectors using the *UseLSX* “*lsxlc” statement in the script.

Terms and Concepts

Metadata

This is a generic term referring to a Connector’s data definition. The data definition includes the names of data elements, their datatypes, and implies the order of the elements. For example, Notes uses ‘forms’ to describe both the names of data fields as well as the data type of each field; Sybase metadata describes a ‘table’. Refer to Appendix C, “Connector Properties”, for information specific to each Connector metadata.

Alternate Metadata

Available through some Lotus Connectors, this is an alternate source for the data definition. For example, DB2 metadata is in the form of a ‘table’, while its alternate metadata form is a ‘SQL view’. Refer to Appendix C, “Connector Properties”, for information on whether a specific Connector supports alternate metadata.

Result Set

The return from an information request through a connection. Each connection can have a single active result set. The LCCONNECTION methods Execute, Select, Call and Catalog each produce a result set, replacing any existing result set. The result set describes the collection of data or information from the connection, which matched the input criteria. It does not return the actual data until fetched. If desired, these methods will build a fieldlist representing the metadata as part of generating the result set.

When using other connection methods which read or write data of a result set, the implied order of the metadata may be suspended by using the connection's `OrderBy` property to indicate that, regardless of the order of the data elements within the metadata, match the names in the metadata to the names in the external system.

Writeback Result Set

A Writeback result set is an optimized form of result set supported by some Connectors. A Writeback result set provides both sequential read and write operations on the data, and may be used for efficient Update and Remove operations by directly operating on the most recently fetched record in the result set, rather than having to locate the information in the external system a second time. Some Connectors may implement locking in the back end for Writeback result sets.

Token

A token is an integer used to identify a property of a connection. All connection properties have a token value. Common properties have predefined tokens represented by constants. Connection-specific properties do not have predefined tokens. (For a list of property tokens and names for each connection, see Appendix C.) Connection properties may be accessed by token or by name. The token method may be used when testing if a property is supported. The name may be used when it is known that a given property exists.

LSX LC Classes

The Lotus Connectors provides external data and system access to the Domino LotusScript environment. The LSX LC classes consist of `LCConnection`, `LCFieldlist` and `LCField`, and four advanced datatypes, `LCStream`, `LCNumeric`, `LCCurrency`, and `LCDatetime`. In addition to these 7 classes, there is also an `LCSession` class. Each of these classes and their primary usage is described below:

LCSession

The `LCSession` class provides error information useful in error handlers. It also provides for query and lookup of available Lotus Connectors.

LCConnection

The `LCConnection` class represents an instance of a Lotus Connector. This class provides query and data access to the external system. Multiple connections may be allocated to a single Connector.

LCFieldlist

The LCFieldlist class is the primary class for manipulating data through a connection. It binds a group of fields together with names and an implied order.

Fieldlists are used primarily for a number of connection operations; result sets and selection criteria, as well as reading and writing data.

When a result set is generated, and an empty fieldlist was initially passed in, the fieldlist is automatically populated by the Connector. For each data element of the result set, the fieldlist receives the element's name and a field object of the corresponding datatype. The result set may be controlled by manually building the fieldlist before the result set is constructed or by using the FieldNames property of the connection.

The Select and Call connection methods use an optional fieldlist of keys or parameters to restrict the result set. This fieldlist is manually constructed and passed to the select method. A key or parameter list is constructed by appending or inserting names and datatypes to the list. These methods create fields in the fieldlist and return these fields for further manipulation. These fields are then given values and, using field flags, may be given conditions such as greater-than, not-equal, etc.

LCField

LCField is the storage class that contains one or more data values. The datatype of a field is for all values contained within and may be any of the four advanced datatypes below, as well as long integer and double precision floating point, and, in some advanced usage, fieldlist or connection.

LCStream

LCStream is a general purpose text and binary datatype. The contents of a stream are marked with a format that details the character set of the text or any special attributes of the binary data.

LCNumeric

The LCNumeric class is a container for very high precision numbers.

LCCurrency

The LCCurrency class is a fixed point decimal datatype with 4 decimal places and 19 digits of precision. (This is mathematically equivalent to the LotusScript datatype and is provided to support connections with a dedicated currency.)

LCDatetime

The LCDatetime class is a date and time datatype which is accurate to the hundredth of a second and which is aware of time zones and daylight savings time.

Working with the LSX

A typical use of the Lotus Connectors is to gather, create, or modify data in an external system. For example, a Notes application has a number of data fields on a form. Once the user input is complete, a button activates script to take the form data and establish a connection to Oracle, locating corresponding information from one or more tables and updating the Notes form. The script is responsible for:

1. accessing the active Note form
2. gathering the input data
3. create a connection to the external system
4. selecting the data based on the input values
5. loading the data from the external system
6. storing the results in the active Notes form

Here is a simple script to accomplish the task. The assumption is that the Notes form has a single text input field called "Customer". The script will use the value of the customer field to location the corresponding customer information in DB2 and return an address and phone number for the customer storing the return values in the form using fields called "Address", "City", "State", and "Phone".

NOTE: No attempt has been made to prescribe a code style. The practice of grouping object declarations together at the beginning of a script versus locating declarations close to the code is a preference and does not affect the execution. In this example, declarations and code are grouped to facilitate explaining the process.

The first step in writing the script is to load the LotusScript Extensions for Lotus Connectors. The UseLSX statement accomplishes this step. Additional Options may be used to check variables, simplify string comparison, etc.

```
Option Public
Option Explicit
UseLSX "*/lsxc"
```

The remainder of the script is located in the 'Click' event of the form's button. Errors should be displayed to the user. A simple error handler is written at the bottom of this example. The LSX

Overview of the LotusScript Extension for Domino Connectors

Session class has a Status property that may be used to determine if the error handler was triggered by an LSX error or a LotusScript error. In all cases where the LSX reports an error, the LotusScript 'Error\$' will contain error information. However, when first creating LSX objects, the LSX has additional error information not available through the LotusScript error statements. Creating and initializing the Session status provides this additional information for the error handler. The creation of the session object is not necessary for normal error handling.

```
Sub Click (Source as Button)
    On Error Goto Handler
    Dim session as New LCSession
    session.ClearStatus
```

The input values are in the current active document. This information is accessible via the NotesUIDocument which may be located through the NotesUIWorkspace from its 'CurrentDocument' property..

```
Dim wksp As New NotesUIWorkspace
Dim uidoc As NotesUIDocument
Set uidoc = wksp.CurrentDocument
```

The next step establishes a connection to the Lotus Connector for DB2. After the connection has been created, all of its properties are accessible to customize the connection to the target system. Common properties include Database and/or Server, UserID, and Password. Properties are not case sensitive. (For a complete list of properties for each Lotus Connector, see Appendix B.) The following code connects to the DB2 system called Rainbow as *jdoe* with the password *gold*.

```
Dim src As New LCConnection ("db2")
src.Database = "Rainbow"
src.UserID = "jdoe"
src.Password = "gold"
src.Connect
```

There are four connection methods for querying through a connection: Catalog, Execute, Select and Call. The catalog operation is used to return metadata information within the external system, for example, the tables of a DB2 database or the columns of a specific Sybase table. For a complete list of Catalog options, see the Catalog method. The remaining methods, Execute, Select and Call, create result sets of data from the connection. The methods differ significantly in their interface. The execute statement uses a Connector-specific command statement to determine the contents of the result set. This interface is helpful when the external system's command structure is familiar and when cross Connector portability is not an issue. The Select method uses a combination of key names, values, and condition flags to indicate the desired contents of the result set. This interface works across Connectors and does not require knowledge of the connector's command language. The Call method is similar to Select, but is used for calling back end procedures or functions. Instead of keys, parameters are provided.

For our example, the important data are stored in the "Customer" DB2 table, as indicated by the Metadata property of the connection. The only record of interest is the customer named by the input value from the Notes form. This selection is accomplished by creating a key list. The default key flag, LCFIELD_KEY, indicates match exactly. If an inequality match such as 'greater-than' or 'like' is needed, then the field's flags property would be ORed with the

Overview of the LotusScript Extension for Domino Connectors

corresponding constant. (In all cases, key fields must have the LCFIELDF_KEY constant in addition to any optional conditional flag constants.)

```
Dim keys As New LCFieldList
Dim field As LCField

src.Metadata = "Customer"
Set field = keys.Append ("Name", LCTYPE_TEXT)
field.Flags = LCFIELDF_KEY
field.Text = uidoc.FieldGetText ("Customer")
```

The Select connection method creates a result set of all records from the external system which match the keylist. If the LotusScript keyword "Nothing" is substituted for the key list, then all records of the specified metadata would be selected. In this case, all records from the "Customer" DB2 table would be selected. This example is interested in just the customer record matching the input value from the Notes form. The key list is created to make this restriction.

The fieldlist receiving the result set is currently empty. The selection will populate the fieldlist with the fields from the DB2 table. If all of the fields of the metadata are not needed, the result set may be restricted to just the fields of interest either by creating the fieldlist prior to the selection or by setting the 'FieldNames' property of the connection.

```
src.FieldNames = "Address, City, State, OfficePhone"
```

The selection returns one of three values: the number of records selected; zero (0) if no matching records were found; or LCCOUNT_UNKNOWN, when records were found but the connection does not know the total. Since a return of zero is the only case where data was not found, it is the test case for error handling or branching.

```
Dim fields As New LCFieldList
If (src.Select (keys, 1, fields) = 0) Then End
```

A result set has been created and there is a match. The result set has not retrieved the data. The Fetch connection method reads the data from the external system into the fieldlist. The individual data values may be accessed from a fieldlist using the expanded class properties. For each field in a fieldlist, there is a property with the corresponding name. This property is an array of values using the closest available LotusScript datatype to match the LSX LC datatypes.

```
If (src.Fetch (fields) > 0) Then
    Call uidoc.FieldSetText ("Address", fields.Address(0))
    Call uidoc.FieldSetText ("City", fields.City(0))
    Call uidoc.FieldSetText ("State", fields.State(0))
    Call uidoc.FieldSetText ("Phone", fields.OfficePhone(0))
End If
```

NOTE: When writing scripts that act on more than one record, it is more efficient to locate the field from within the fieldlist, outside the loop, and then use the field for data access. Using the expanded class properties locates the field each time it is used and allocates an array of values, not just a single value. Here is an sample of this code. The %REM has been used to indicate that this code is not part of the actual example.

```
%REM
```

Overview of the LotusScript Extension for Domino Connectors

```
Dim address as LCField
Dim city as LCField
Dim state as LCField
Dim phone as LCField

Set address = fieldlist.Lookup ("Address")
Set city = fieldlist.Lookup ("City")
Set state = fieldlist.Lookup ("State")
Set phone = fieldlist.Lookup ("OfficePhone")

while (src.Fetch (fields) > 0) Then
    Call uidoc.FieldAppendText ("Address", address.Text(0))
    Call uidoc.FieldAppendText ("City", city.Text(0))
    Call uidoc.FieldAppendText ("State", state.Text(0))
    Call uidoc.FieldAppendText ("Phone", phone.Text(0)) End If
%END REM
```

The data has been retrieved from the external system and placed in the Notes form. This completes this example. The final step is to refresh the Notes document to display the new data to the user.

```
uidoc.Refresh
End
```

An error handler was designated as the first line of this example. Testing for an LSX error first provides additional information in the case of an object creation error. Without the session object and subsequent test in the error handler, failure while creating a connection to DB2 generates the LotusScript message, "Error creating product object". However, for the same error condition, the LSX reports "Error: Cannot load LSX library 'db2'".

```
Handler:
    If (Session.Status <> LCSUCCESS) Then
        Messagebox Session.GetStatusText, 0, "The following Lotus Connector error
has occurred"
    Else
        Messagebox Error$, 0, "The following LotusScript error has occurred"
    End If
End
End Sub
```

This example is very simple. It is meant only to provide an understanding of the Lotus Connectors, the classes, and the relationship between the connection, metadata result set, fieldlist, fields and data.

LSX LC Usage Notes

Environment

Use of the Lotus Connectors LotusScript Extensions is limited to the Notes environment and Notes applications. For example, LC LSX scripts cannot be called from within Lotus 1-2-3 or Lotus Approach, because the Connectors only have context within the Domino Server and Notes client.

LSX LC Data Types

Standard Data types

Standard data types come in three classes: Number (Int, Float, Currency, and Numeric), Datetime (Datetime), and Stream (Text and Binary). All types except Int and Float are represented by Lotus Connector classes described in this manual (Int is represented by a LotusScript Long, and Float by a LotusScript Double).

- Int: A four-byte signed integer. Integers are very efficient, but limited in precision and range. An int is any whole number between -2147483647 and 2147483647 (9 digits of precision).
- Float: An eight-byte (double-precision) floating point number. Floats have less precision than currencies or numerics, but have a far greater range of values and are more efficient. A float has 15 digits of precision, and can represent any practical value (with an exponential range between 10^{308} and 10^{-308}).
- Currency: An eight-byte integer with four decimal places. Currency provides greater precision than either int or float values. While it is less precise than a numeric, it is far more efficient, making it the preferable choice when higher precision is required. A currency has 19 digits of precision, and a maximum value of 922,337,203,685,477.5807
- Numeric: A numeric is larger and less efficient than any other number datatype, but has far greater precision. A numeric is assigned a precision (number of digits) up to 88 and a scale (number of decimal places) between -127 and 127, allowing it to express a very large number of values with very high precision.
- Datetime: A datetime indicates a specific date and time. Its range is any date between the years 1 and 32767, with a precision of .01 seconds. In addition, datetimes are specific to timezone and DST settings, making them usable in international settings.

- **Text:** Text is a stream of characters up to 4 Gb in length. All text has a character set, and conversion between character sets is supported. When transferring between systems, character set conversion is automatically performed as needed.
- **Binary:** Binary is a stream of bytes up to 4 Gb in length. All binary data has a format which is either BLOB (Binary Large Object) for unformatted data, or a structured binary format such as composite, text list, number list, and datetime list. The three list formats allow for a single data value to itself contain multiple values.

Advanced Data types

Fieldlist: A fieldlist itself is a collection of named fields with one or more values each, similar to a SQL table. A fieldlist as a data type allows for a single data value to itself contain a fieldlist, supporting hierarchical or nested data. The fields in a fieldlist data value may in turn contain additional nested fieldlists. Fieldlists are only usable by those back end that support such data structures.

Connection: A connection supports interaction with a back end system through a Lotus Connector. Connections as a data type are only valid when working with metaconnectors, since metaconnectors have properties of type Connection.

Arrays and Indexes

The default for array indexes in LotusScript is 0 based. For example, elements of an array are specified as Elem(0), Elem(1), Elem(2), etc. There is an option in LotusScript, "Option Base n", that makes the default lower bound of an unspecified array something other than 0. LSXs do not support this syntax. It is recommended, for consistency, that within scripts which use an LSX, such as the Lotus Connectors, that this option not be used.

Many Lotus Connector class methods operate on multi-value data. Class methods include the 'List' methods of LCStream, the 'Get' and 'Set' methods of LCField and LCFieldlist, as well as the 'Fetch', 'Insert', 'Update', and 'Remove' methods of LCCConnection. These operations are counted from 1. For example, fields of a fieldlist are specified as MyFieldList.GetField (1), MyFieldList.GetField (2), MyFieldList.GetField (3), ... etc. Unlike dimensioned arrays which default to zero based, and LSX generated arrays which are always zero based, these class methods require an index counted from 1.

It is possible to assign a LotusScript array to a multi-value object such as a LCField. It is important that the array not contain more elements than the LCField object will store as this will cause an overflow error. For example, 5 long integers may be assigned to an LCField as follows:

```
Dim numbers(4) as Long ' numbers(0, numbers(1), ... numbers(4)
Dim field as new LCField (LCTYPE_INT, 5)
...
field.Value = numbers
```


Working with Multiple Rows of Data

The Fetch, Insert, Update, Remove methods of a connection use a *RecordCount* parameter and may operate on more than a single row at a time. However, the *LCFieldlist* parameter must have been created to hold more than one record for this to work. You may operate on less records than the *Fieldlist* was created with, but not more.

A simple example of different configurations for *RecordIndex* and *RecordCount* might be as follows:

```
Dim Records as New LCFieldlist (3, LCFIELDF_TRUNC_DATA)
```

```
...
```

```
Call connection.Fetch (Records, 1, 3)      ' fetch three records
```

```
Call connection.Insert (Records, 3, 1)     ' insert them in reverse order
```

```
Call connection.Insert (Records, 2, 1)
```

```
Call connection.Insert (Records, 1, 1)
```

Note that both parameters for New *LCFieldlist* are optional and default to 1, and *LCFIELDF_TRUNC_PREC*, respectively. Likewise, the *RecordIndex* and *RecordCount* parameters are optional for Fetch, Insert, Update and Remove, and default to 1 and 1. Lastly, the *RecordIndex* and *RecordCount*, when added to each other (minus 1) must not exceed the size of the created *Fieldlist*.

Optional Parameters

Some methods have optional parameters. Note the LotusScript keyword "Nothing" is used for output parameters that the user is not interested in receiving. If one or more optional parameters are at the end of a call, they may simply be ignored as in the case of a datetime. For example, a datetime with only the date portion provided is created using

```
Dim Calendar as New LCDatetime (1998, 8, 7)
```

When optional parameters that occur in the middle are omitted, the commas must remain, as in:

```
Call session.GetStatus (ErrorText, , DBError)
```

Performance

Certain techniques can be used to increase LSX LC code performance. These techniques are described below.

Connector Caching

Most connectors are optimized for repeated calls using the same parameters, but different data values. This generally applies to Fetch, Insert, Update, Remove, and Select operations. When these methods are called with the same fieldlist more than once, they are able to avoid certain steps such as field mapping and type checking. This optimized functionality is available as follows:

Select: METADATA property and KeyFieldlist are the same.

Fetch: Result set and DestFieldlist are the same.

Insert: METADATA property and SrcFieldlist are the same.

Update and Remove: WRITEBACK and METADATA properties and SrcFieldlist are the same.

Calls to one type of method generally do not affect others, so the following pseudo-code obtains the optimal use of this caching:

```
Connect to Connection CONN
Set Metadata
Generate result set, producing Fieldlist FL
Loop until done:
    Fetch into FL
    Insert from FL
Exit
```

If this were changed to fetch the data and then insert into one of two tables depending on some condition, then every time the table changed, new field mapping and type checking would be performed. In such a case, establishing two connections for Insert would be optimal.

Tight Loops

Another way to optimize LSX LC code is to produce tight loops, moving as much code out of the loop as possible. Anything inside such a loop must be done for every iteration of the loop, and can therefore become costly with large data sets. For example, inside a loop which transfers 10,000 records, you may need to access a field in a fieldlist. Instead of using the property as Fieldlist.Fieldname, which will locate the field by name every iteration, you could instead use Field = Fieldlist.Fieldname outside the loop and then simply refer to Field inside the loop. This will remove 10,000 property accesses for the fieldlist.

NOTES.INI Variables

The following entries may be added to the NOTES.INI file to control aspects of the LSX LC:

DECSTranslation: This controls text translation, allowing the user to increase performance in exchange for certain assumptions about the data being accessed. Note that none of these settings affects translation between unicode and other character sets, since it is always required. There are three valid numeric settings:

- 0 Do not perform translation between character sets (except unicode). This is valid when all data being accessed is compatible with the Notes LMBCS character set - primarily ASCII printable characters.
- 1 No not performed translation between non-LMBCS (and non-unicode) character sets. This is valid when all data being accessed, except for Notes LMBCS data, is in compatible character sets.
- 2 Always translate between any character sets. This is the default.

DECSCenturyBoundary: This controls how to interpret the year in a text string being converted to a datetime, when the year contains only two digits. Values greater than or equal to the century boundary are considered to be in 1900s, values less than are in the 2000s. There are three ways to set this:

- 0 Since all values are greater than or equal to zero, always use 1900.
- 1-100 If the two-digit year is greater than or equal to this value, use 1900; otherwise use 2000.
- 101 Since all values are less than 101, always use 2000.

The default setting is 50, which is the same as Notes. This means than any two-digit year from 0-49 is in 2000, and 50-99 is in 1900.

DECSNativeText: This allows the local machine's native character set to be overridden. There are various situations when the native character set is used within the LSX LC, and some backend systems always consider client data to be in the native character set. Setting this value to a valid text format string replaces the character set obtained from the operating system by the LSX LC with the indicated character set. Appendix D lists supported character sets. Use the text that remains after removing the "LCSTREAMFMT_" prefix. For example, Code Page 932, represented by the constant LCSTREAMFMT_IBMCP932, would be set as follows:

DECSNativeText=IBMCP932.

Chapter 2

LCConnection Class

This chapter describes the LCConnection class, and its methods and properties. Each method and property is listed alphabetically.

Overview

The LCConnection class represents an instance of an individual Lotus Connector. One connection object should exist for each individual data connection through a connector. This is true when the connections are to the same Lotus Connector as well as to different connectors. For example, transferring data from one DB2 table to another table in Oracle is best accomplished by creating two LCConnection objects, one for DB2 and one for Oracle, and fetching data from one and inserting it into the other.

Connection methods manage individual Lotus Connector connections within a session. The Connection class enforces Connector state and requirements, and must always be used when interacting with Connectors. One connection object should exist for each individual data connection through a Connector used in a session.

Error information for a connection is available through use of the LCSession class.

LCConnection Properties

Connection properties are specific to the Lotus Connector. Refer to Appendix A for a list of properties for each Lotus Connector. All connector properties are of the closest LotusScript datatypes.

Properties of a connection maybe accessed by name, for example, the following line of script sets the value of Database property for a DB2 connector to "HR":

```
Connection.Database = "HR"
```

LCConnection Class Methods Summary

Connection

The following methods control connection to a data provider. A connection is required before gaining access to most Connector metadata and all Connector data. Multiple connections may be established to a single Connector with multiple Connection objects.

LCConnection.Connect Establish a connection to a data provider.

LCConnection.Disconnect Disconnect from a data provider.

Create Result Set

Each Connection can have a single active result set. A result set is the data produced by an action, for example, the execution of a Select statement against SQL tables. All of these methods produce a result set, replacing any existing result set. The result set can be produced from a Connector-specific language statement, from Connector-independent properties and keys or parameters, or from Connector metadata. Connector metadata is either an SQL table or view, a Notes form, a BEA Tuxedo service, etc. Once a result set is produced the data in that result set can be fetched. Under specific circumstances, efficient writeback updates and removes directly back into the result set are also supported.

Note that the Execute and Call methods invoke an operation from the external system.

LCConnection.Call Perform a Connector-independent procedure call with parameters.

LCConnection.Catalog Produce a result set containing a metadata catalog

LCConnection.Execute Execute a statement against the data provider. The statement is provided in the data provider's language.

LCConnection.Select Perform a Connector-independent selection controlled by various Connector properties. Conditional key inequalities, timestamp, and other control is supported.

Data Manipulation

These methods allow access to and manipulation of Connector data.

<code>LCConnection.Fetch</code>	Retrieve records from the current result set.
<code>LCConnection.Insert</code>	Insert new records into the data provider.
<code>LCConnection.Update</code>	Update records in the data provider. Key values and update values are provided. No keys are required for writeback operations.
<code>LCConnection.Remove</code>	Delete records from the data provider. Key values are provided. No keys are required for writeback operations.

Metadata Manipulation

These methods allow access to and manipulation of Connector metadata.

<code>LCConnection.Create</code>	Create a new metadata object.
<code>LCConnection.Drop</code>	Drop an existing metadata object.

Miscellaneous

<code>LCConnection.Action</code>	Perform one of a set of predefined actions.
<code>LCConnection.GetProperty</code>	Fetch a property value for a connection.
<code>LCConnection.GetProperty<datatype></code>	Fetch a property as a particular data type.
<code>LCConnection.ListProperty</code>	List supported properties and values.
<code>LCConnection.LookupProperty</code>	Verify the support of a property.
<code>LCConnection.SetProperty</code>	Set a property value for a connection.
<code>LCConnection.SetProperty<datatype></code>	Set a property as a particular data type.

Connector Properties

Refer to Appendix C, "Connector Properties," for more information about the properties for each Lotus Connector.

New method for LCConnection

This is the constructor for objects of class LCConnection.

Defined In

LCConnection

Syntax

Dim *connectionName* as **New** LCConnection(*libraryName*)

Parameters

libraryName

The name of a valid, installed Connector, such as "db2" or "oracle". Use lowercase letters for *libraryName*, as some file systems (for example, UNIX) are case-sensitive. See the LCSession.ListConnectors method, which is used to determine the installed Connectors.

Usage Notes

A connection library is located by name.

The constructor allocates a code for this connection unique among all connections, which can be used as a virtual code for a field (see LCField..SetVirtualCode). This value can be retrieved as the Connector property LCTOKEN_CONNECTION_CODE.

Action method for LCConnection

This method performs an action as defined by the *actionType* parameter.

Defined In

LCConnection

Syntax

Call *lcConnection.Action(actionType)*

Parameters

actionType

Long. One of the following values:

LCACTION_RESET

Returns the Connector to a state equivalent to that just after connection. All outstanding results are committed and all result sets and state information are freed. Supported by all Connectors.

LCACTION_TRUNCATE

Deletes all records in the property METADATA in the most efficient available manner determined by the Connector.

LCACTION_COMMIT

Commits all changes in the current transaction. Only supported by Connectors with transaction functionality.

LCACTION_ROLLBACK

Rolls back all changes in the current transaction. Only supported by Connectors with transaction functionality.

LCACTION_CLEAR

Clears the current result set, freeing any locks, but does not affect any other context.

Example

```
Option Public
Option Explicit
Uselsx "**!sxlc"
```

```
Sub Initialize
    Dim session As New LCSession
    Dim src As New LCConnection ("db2")

    REM set properties to connect to both data sources
    src.Database = "Gold"
    src.Userid = "JDoe"
    src.Password = "xyzyz"
    src.Metadata = "customer"
    REM now connect
    src.Connect

    ' check to see if the target metadata exists, if so clear it out.

    ' check for LCFAIL_INVALID_METADATA error
    On Error LCFAIL_INVALID_METADATA Goto NoMetadata
    Call src.Action (LCACTION_TRUNCATE)
    Print "the table '" & src.Metadata & "' has been truncated."
    Print "This removed all existing records."
    End

NoMetadata:
    ' couldn't truncate the table since it didn't exist
    Print "the table '" & src.Metadata & "' does not exist."
    End
End Sub
```

Example Output

```
the table 'customer' has been truncated.
This removed all existing records.
```

Call method for LCConnection

This method is used to call a stored procedure and potentially produce a result set.

This method only supports input parameters to the stored procedure. If you want data returned from a stored procedure, it must be returned to a result set, not by output parameters.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Call**(*parmFieldList*, *recordIndex*, *destFieldList*)

Parameters

<i>parmFieldList</i>	LCFieldlist. The input parameter list for the stored procedure.
<i>recordIndex</i>	Long. The index location of the parameter values within the fieldlist.
<i>destFieldList</i>	LCFieldlist. Fieldlist to contain the metadata of the result set. The fields in the result set will be appended to this fieldlist. If the result set metadata is not required, use Nothing.

Return Value

<i>count</i>	The number of records affected by the call. Note that not all data sources return a <i>count</i> . LCCOUNT_UNKNOWN is returned if the <i>count</i> is not determined.
--------------	---

Example

```
Option Public  
Uselsx "**Isxlc"
```

```
Sub Initialize  
    Dim Con As New LCConnection ("sybase")  
    Dim Parms As New LCFieldList  
    Dim Result As New LCFieldList  
    Dim Parm As LCField  
  
    ' set properties to connect to both data sources  
    Con.Server = "Rainbow"  
    Con.Userid = "JDoe"  
    Con.Password = "xyzyzy"  
  
    ' set the connection property to the stored procedure name  
    Con.Procedure = "NPInsertIntoSM_ADBOOK"  
  
    ' now connect  
    Con.Connect  
  
    ' append the new field to the fieldlist  
    Set Parm = Parms.Append ("spParm", LCTYPE_TEXT)  
    ' set the field to a value - in this case it is  
    ' the one parameter needed for the stored procedure  
    Parm.text = " 'Edge' "  
  
    ' using the fieldlist containing the field with the  
    ' stored procedure parameter, call the stored procedure  
    If (Con.Call (Parms, 1, Result) = 0) Then  
        Print "No results were generated from the procedure call."  
    Else  
        Print "A result set was generated from the procedure call."  
    End If  
End Sub
```

Example Output

A result set was generated from the procedure call.

Catalog method for LCConnection

This method catalogs through metadata related information.

Any active result set for this connection will be replaced. Different metadata types may be cataloged, although all Connectors may not support all object types. The format of the result set produced is returned in the supplied fieldlist, and the result set contents can be retrieved with `LCConnection.Fetch`.

A connection is not required for Server and Database Catalogs for some Connectors. This is an exception, as a connection is required for all other result sets.

Defined In

LCConnection

Syntax

count = *lcConnection.Catalog(objectType, destFieldlist)*

Parameters

objectType

Long. Type of metadata information to be cataloged. Use an `LCOBJECT_XXX` constant to define the metadata type. The following list gives the required context and the resulting metadata format for each catalog type. The general fields, Name, Owner, and Comment, are produced for all objects, but only the first output field(name) is guaranteed to have data. All output fields are data type Text, unless otherwise specified. Field cataloging adds a fourth element, datatype.

`LCOBJECT_SERVER`

Context: None

Output Fields: Server Name, Server Owner, Server
Comment

Catalog method for LCConnection

LCOBJECT_DATABASE

Context: SERVER property (if supported by this Connector)

Output Fields: Database Name, Database Owner,
Database Comment

LCOBJECT_METADATA

Context: Current connection, ALTERNATE METADATA
property

Output Fields: Metadata Name, Metadata Owner,
Metadata Comment

LCOBJECT_INDEX

Context: Current connection

Output Fields: Index Name, Index Owner, Index
Comment

LCOBJECT_FIELD

Context: Current connection, METADATA property,
ALTERNATE METADATA property

Output Fields: Field Name, Field Owner, Field
Comment, Field Datatype Constant

Value	Constant	Type
0	LCTYPE_INVALID	Unknown
1	LCTYPE_INT	Integer
2	LCTYPE_FLOAT	Float
3	LCTYPE_CURRENCY	Currency
4	LCTYPE_NUMERIC	Numeric
5	LCTYPE_DATETIME	Datetime
6	LCTYPE_TEXT	Text
7	LCTYPE_BINARY	Binary
8	LCTYPE_FIELDLIST	Fieldlist
9	LCTYPE_CONNECTION	Connection

destFieldlist

LCFieldlist. Output. Fieldlist to contain the metadata of the catalog result set. The fields in the result set will be appended to this fieldlist. If the result set metadata is not required, use Nothing.

Return Value

count

Long. Number of catalog records available in the result set produced. This value is LCCOUNT_UNKNOWN if the number of records cannot be determined by the Connector.

Example

```
Option Public
Uselsx "**!sxlc"
```

```
Sub Initialize
```

```
  Dim connect As New LCConnection ("db2")
  Dim conFldLst As New LCFieldList
  Dim field As LCField
```

```
  ' this section assigns the appropriate properties to connect to DB2
  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "xyzzzy"
  connect.Metadata = "customer"
```

```
  ' connect to DB2
  connect.Connect
```

```
  ' now perform the catalog action - in this case for metadata
  If (connect.Catalog (LCOBJECT_FIELD, conFldLst) = 0) Then
    Print "No tables were found."
```

```
  Else
```

```
    ' fetch the results
```

```
    Set field = conFldLst.GetField(1)
```

```
    Print "The list of columns in the " & connect.Metadata & _
    " table include:"
```

```
    While (connect.Fetch (conFldLst) > 0)
```

```
      Print "    " & field.text(0)
```

```
    Wend
```

```
  End If
```

```
End Sub
```

Example Output

The list of columns in the 'CUSTOMER' table include:

ACCOUNTMANAGER
CONTACTNAME
COMPANYNAME
COMPANYADDRESS
COMPANYCITY
COMPANYSTATE
COMPANYPHONE

Connect method for LCConnection

This method establishes a connection to a Connector. Multiple connections may be independently established to a single Connector.

Defined In

LCConnection

Syntax

lcConnection.Connect

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim connect As New LCConnection ("db2")  
  
    ' set the appropriate properties to connect to DB2  
    ' note the use of dynamic properties to do this  
    ' all properties of a connection may be referenced  
    ' by name  
    connect.Database = "Gold"  
    connect.Userid = "JDoe"  
    connect.Password = "xyzyz"  
  
    REM try the connect  
    connect.Connect  
    Print "Successfully connected to DB2."  
End Sub
```

Example Output

Successfully connected to DB2.

Copy method for LCConnection

This method makes a copy of an existing connection, including all property values. Note that while all properties are copied, the current state of the connection and result set are not copied.

Defined In

LCConnection

Syntax

Set *newConnection* = *lcConnection*.**Copy**()

Parameters

<i>lcConnection</i>	The connection object that you want to copy.
---------------------	--

Return Value

<i>newConnection</i>	The copy of the <i>lcConnection</i> object.
----------------------	---

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim session As New LCSession
    Dim connect1 As New LCConnection ("db2")
    Dim connect2 As LCConnection
```

```
    ' set the appropriate properties to connect to DB2
```

```
    ' note the use of dynamic properties to do this
```

```
    connect1.Database = "Gold"
```

```
    connect1.Userid = "JDoe"
```

```
    connect1.Password = "xyzzz"
```

```
    ' now copy the connect
```

```
    Set connect2 = connect1.Copy
```

```
    If (connect2.Database = connect1.Database) Then
```

```
        Print "The copy of connection has the same database as the original."
```

```
    Else
```

```
        Print "The copy of connection has a different database from the original."
```

```
    End If
```

```
End Sub
```

Example Output

The copy of connection has the same database as the original.

Create method for LCConnection

This method creates a metadata object. Each Connector supports only certain object types. Refer to the documentation for the specific Connector to determine the object types it supports.

Defined In

LCConnection

Syntax

Call *lcConnection.Create(objectType, srcFieldlist)*

Parameters

objectType

Long. Type of object. Valid types and associated behavior include the following (refer to the documentation for the specific Connector to determine which of the following are supported):

LCOBJECT_SERVER

Creates a server object (obtaining the name from the SERVER property). Additional information may be provided in Connector-specific properties.

LCOBJECT_DATABASE

Creates a database object (obtaining the name from the DATABASE property, and optionally the server name from the SERVER property). Additional information may be provided in Connector-specific properties.

LCOBJECT_METADATA

Creates a metadata object (obtaining the name from the METADATA property). The fields in the new metadata will have the same order, types, and names as fields in the fieldlist. Fields with the flag LCFIELDF_NO_CREATE are skipped.

LCOBJECT_INDEX

Creates an index object (the metadata being indexed is in METADATA property, the index name to create is in INDEX property). The key fields for the new index are fields in the fieldlist with the LCFIELD_KEY flag set.

LCOBJECT_FIELD

Creates a field object (the metadata being appended to is in METADATA property). The fields to append to the metadata will have the same order, types, and names as fields in the fieldlist. Fields with the flag LCFIELDF_NO_CREATE are skipped.

srcFieldlist

LCFieldlist. Fieldlist from whose metadata or key fields the object is created. This parameter is ignored for object types SERVER and DATABASE.

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

Sub Initialize

```
Dim src As New LCConnection ("db2")
Dim fldLstRecord As New LCFieldList
Dim fld As LCField

' build the table definition
' note the use of the 'MaxLength' parameter
' this is used to more closely control the datatype creation
' within the connection - in this case DB2
' in DB2, a text stream with a MaxLength of 64 will be created
' as VARCHAR(64). if the flag LCSTREAMF_FIXED was used
' the column would be CHAR(64). The field flag LCFIELDF_NO_NULL
' would add NOT NULL to the column definition

Call fldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
Set fld = fldLstRecord.Append ("CONTACTNAME", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = fldLstRecord.Append ("COMPANYNAME", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = fldLstRecord.Append ("COMPANYADDRESS", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = fldLstRecord.Append ("COMPANYCITY", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = fldLstRecord.Append ("COMPANYSTATE", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = fldLstRecord.Append ("COMPANYPHONE", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 32, LCSTREAMFMT_LMBCS)

' set properties to connect to both data sources
src.Database = "Gold"
src.Userid = "JDoe"
```

Create method for LCConnection

```
src.Password = "xyzzzy"  
src.Metadata = "customer"  
  
src.Connect  
  
' create it based on the metadata property already set above  
On Error LCFAIL_DUPLICATE Goto tableexists  
Call src.Create (LCOBJECT_METADATA, fldLstRecord)  
Print "The '" & src.Metadata & "' table was created."  
End  
  
tableexists:  
Print "The '" & src.Metadata & "' table exists."  
End  
End Sub
```

Example Output

The 'customer' table was created.

Disconnect method for LCConnection

This method disconnects from a data source. Any existing result set is cleared.

Defined In

LCConnection

Syntax

lcConnection.**Disconnect**

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
  Dim connect As New LCConnection ("db2")

  ' set appropriate properties to connect to DB2
  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "DBINST1"

  ' connect to DB2 then disconnect
  connect.Connect
  Print "Successfully connected to DB2."

  ' now lets disconnect
  connect.Disconnect
  Print "Successfully disconnected from DB2."
End Sub
```

Example Output

```
Successfully connected to DB2.
Successfully disconnected from DB2.
```

Drop method for LCConnection

This method drops the specified object type.

Each Connector supports only certain object types. Refer to the documentation for the specific Connector to determine the object types it supports.

Defined In

LCConnection

Syntax

Call *lcConnection.Drop(objectType)*

Parameters

objectType

Long. Type of object: Valid types and associated behavior are the following (refer to Appendix C for the specific Connector to determine which of the following are supported):

LCOBJECT_SERVER

Drops a server object (obtaining the name from the SERVER property).

LCOBJECT_DATABASE

Drops a database object (obtaining the name from the DATABASE property, and optionally the server name from the SERVER property).

LCOBJECT_METADATA

Drops a metadata object (obtaining the name from the METADATA property).

LCOBJECT_INDEX

Drops an index object (the metadata indexed is in METADATA property, the index name to drop is in INDEX property).

LCOBJECT_FIELD

Drops a field object (metadata containing fields is in METADATA property, the fields being removed are in the FIELD_NAMES (or FieldNames) property).

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
  Dim src As New LCConnection ("db2")
  ' set properties to connect to the data source
  src.Database = "Gold"
  src.Userid = "JDoe"
  src.Password = "xyzzzy"
  src.Metadata = "customer"

  src.Connect

  On Error Goto NoMetadata
  Call src.Drop (LCOBJECT_METADATA)
  Print "The '" & src.Metadata & "' table existed and had been deleted."
  End

NoMetadata:
  Print "The '" & src.Metadata & "' table did not exist."
  End
End Sub
```

Example Output

The 'customer' table existed and had been deleted.

Execute method for LCConnection

This method executes a statement in Connector-specific syntax, for example, an SQL statement for a relational database Connector.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Execute**(*statement*, *destFieldlist*)

Parameters

<i>statement</i>	String. The statement to execute in the syntax defined for the Connector. See the documentation for the specific Connector for information about the required syntax.
<i>destFieldList</i>	LCFieldlist. Fieldlist to contain the metadata of the execute result set. The fields in the result set will be appended to this fieldlist. If the result set metadata is not required, use Nothing.

Return Value

<i>count</i>	Number of records selected or affected by this statement. If this number cannot be determined by the Connector, the constant LCCOUNT_UNKNOWN is returned.
--------------	---

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
  Dim src As New LCConnection ("db2")
  Dim fldLst As New LCFieldList
  Dim fld As LCField
  Dim count As Integer
```

```
  ' set the appropriate properties to connect
```

```
  src.Database = "Gold"
  src.Userid = "JDoe"
  src.Password = "xyzyzy"
```

```
  src.Connect
```

```
  ' now connected, we can execute a selection statement
```

```
  If (src. Execute ("SELECT * from customer", fldLst) = 0) Then
    Print "No records were fetched."
  End
```

```
End If
```

```
Set fld = fldLst.Lookup ("CONTACTNAME")
```

```
Print "the 'contact names' stored in the table are:"
```

```
  ' fetch each record from the result set
```

```
  While (src.Fetch (fldLst) > 0)
    count = count + 1
    Print "    record #" & Cstr(count) & " = '" & fld.text(0) & "'"
  Wend
```

```
  If (count = 0) Then Print "No records were fetched."
```

```
End Sub
```

Example Output

the 'contact names' stored in the table are:

```
record #1 = 'Peter Pan'
record #2 = 'R. U. Happy'
record #3 = 'Issac Bernard Mathews'
```

Fetch method for LCConnection

This method obtains the next group of records from a result set. This method requires an active result set in the Connection.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Fetch**(*destFieldlist*, *recordIndex*, *recordCount*)

Parameters

<i>destFieldlist</i>	LCFieldlist. Fieldlist to receive the data. For each field in Fieldlist without the flag LCFIELD_NO_FETCH, data from the corresponding field in the result set will be copied into that field. Fields in the result set and fieldlist are matched by name if the MapByName property is TRUE, by position otherwise, and are type-checked before retrieving data.
<i>recordIndex</i>	Long. Optional. Starting record index in the fieldlist where the record will be stored. Default is 1.
<i>recordCount</i>	Long. Optional. Number of records to fetch. The number of records actually fetched may be less than this number if the end of the result set was reached. While all Connectors can fetch multiple records, only Connectors which indicate support for Array Fetch perform a true multi-record fetch and therefore reduce network traffic and increase performance. Default is 1.

Return Value

<i>count</i>	Long. Number of records successfully fetched.
--------------	---

Usage Notes

You can achieve optimal performance by using the same fieldlist across consecutive fetches from the same target.

Example

Option Public
Uselsx "**Isxlc"

Sub Initialize

Dim src As New LCConnection ("db2")

Dim fldLst As New LCFieldList

Dim keyLst As New LCFieldList

Dim fld As LCField

Dim count As Integer

' set the appropriate properties to connect to the data source

src.Database = "Gold"

src.Userid = "JDoe"

src.Password = "xyzzzy"

src.Metadata = "customer"

src.Connect

' the FIELDNAMES property of a connection is used to

' specify which fields should be used in the resultset

' if no names are listed, then all fields will be fetched

src.FieldNames = "ContactName, AccountManager"

' the select statement may be called with 'Nothing' as

' the keylist parameter. this causes all records to be

' selected for the result set.

' by creating a keylist with one or more keys, conditions,

' and values, tighter control of the result set is possible

' here we want to indicate all account managers except

' number 200

' NOTE: to indicate that a field is a key, the LCFIELDF_KEY flag

' must always be included in the value of the connection's flags

Set fld = keyLst.Append ("ACCOUNTMANAGER", LCTYPE_INT)

fld.Flags = LCFIELDF_KEY_NE Or LCFIELDF_KEY

fld.Value = 200

' the selection statement builds an internal result set which

' is later accessed with successive fetches

If (src.Select (keyLst, 1, fldLst) = 0) Then

Print "No data were located."

End

End If

Set fld = fldLst.Lookup ("CONTACTNAME")

```
Print "the 'contact names' stored in the table are:"  
REM fetch a record from the result set  
While (src.Fetch (fldLst) > 0)  
    count = count + 1  
    Print "    record #" & Cstr(count) & " = '" & fld.text(0) & ""  
Wend  
If (count = 0) Then Print "The table contains no records."  
End Sub
```

Example Output

```
the 'contact names' stored in the table are:  
    record #1 = 'Peter Pan'  
    record #2 = 'R. U. Happy'  
    record #3 = 'Issac Bernard Mathews'
```

GetProperty method for LCConnection

This method retrieves a copy of the current value for a connection property. Note that use of dynamic properties is more efficient.

Defined In

LCConnection

Syntax

Call *thisConnection*.**GetProperty**(*propertyToken*, *destField*)

Parameters

<i>propertyToken</i>	Long. Token representing the Connector property for which the value is returned. See Appendix B for a list of tokens.
<i>destField</i>	Current value for the Connector property. If the property value is of a different type than this field, then data conversion will occur, if possible. The property value will be written into the first value in the field.

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
    Dim connect As New LCConnection ("db2")
    Dim conFld As LCField

    ' get the value of the server property
    Set conFld = connect.GetProperty (LCTOKEN_WRITEBACK)
    Print "The writeback property value is: " & conFld.text(0)
End Sub
```

Example Output

The writeback property value is: 0

GetProperty<Type> Methods

This method returns a Connector property value as a specific data type. Datatypes supported include: Boolean, Currency, Datetime, Float, Int, Numeric, and Text.

This method allows retrieval of Connect properties as with `LCCConnection.GetProperty`, but does not require a field object. If the property value is of a different type, data conversion will be performed. The value of the password property cannot be obtained with this function, and will result in an `INVALID_PROPERTY` error.

Defined In

`LCCConnection`

Syntax

flag = *lcConnection*.**GetPropertyBoolean**(*propertyToken*, *default*)

Set *destCurrency* = *lcConnection*.**GetPropertyCurrency**(*propertyToken*)

Set *destDatetime* = *lcConnection*.**GetPropertyDatetime**(*propertyToken*)

destFloat = *lcConnection*.**GetPropertyFloat**(*propertyToken*)

destInt = *lcConnection*.**GetPropertyInt**(*propertyToken*)

Set *destNumeric* = *lcConnection*.**GetPropertyNumeric**(*propertyToken*)

Set *destStream* = *lcConnection*.**GetPropertyStream**(*propertyToken*, *streamFormat*)

Parameters

<i>propertyToken</i>	A token identifying the Connection property. See Appendix B for a list of property tokens.
<i>default</i>	(GetPropertyBoolean only) Value to be returned if the property cannot be located. Default value is FALSE.
<i>streamFormat</i>	(GetPropertyStream only) Format that the stream is converted to before being returned. If <i>streamFormat</i> is zero, no conversion occurs and the stream is copied in the same format as the property value.

Return Values

<i>flag</i>	GetPropertyBoolean only. Boolean value, either TRUE or FALSE. If the property does not exist, the <i>default</i> is returned.
<i>dest<Type></i>	Current value for the Connector property.

GetProperty<Type> Examples

Option Public

Uselsx "**Isxlc"

Sub Initialize

Dim connect As New LCConnection ("oracle")

Dim conFld As LCField

Dim propName As String

Dim propDate As LCDateTime

Dim propNumeric As LCNumeric

Dim propStrm As LCStream

Dim propCurr As LCCurrency

Dim propFloat As Double

Dim propInt As Long

Dim propBool As Variant

Dim tokenId As Long

Dim propType As Long

Dim propFlags As Long

' set some connector properties

connect.Server = "Rainbow"

connect.Userid = "JDoe"

connect.Password = "xyzzzy"

' it is not necessary to connect to list properties

Call connect.ListProperty (LCLIST_FIRST, _

tokenId, propType, propFlags, propName)

Print "NAME" Tab(20); "ID"; Tab(28); "FLAGS"; _

Tab(36); "TYPE"; Tab(48); "VALUE"

Print "-----" Tab(20); "-----"; Tab(28); "-----"; _

Tab(36); "-----"; Tab(48); "-----"

Do

Set conFld = connect.GetProperty (tokenId)

' match the property to a datatype and fetch it as that datatype

Select Case propType

Case LCTYPE_DATETIME:

Set propDate = connect.GetPropertyDatetime (tokenId)

Print propName; Tab(20); Hex(tokenId); Tab(28); Hex(propFlags); _

Tab(36); "LCDatetime"; Tab(48); propDate.text

Case LCTYPE_NUMERIC:

Set propNumeric = connect.GetPropertyNumeric (tokenId)

Print propName; Tab(20); Hex(tokenId); Tab(28); Hex(propFlags); _

Tab(36); "LCNumeric"; Tab(48); propNumeric.text

Case LCTYPE_TEXT:

Set propStrm = connect.GetPropertyStream (tokenId, LCSTREAMFMT_NATIVE)

Print propName; Tab(20); Hex(tokenId); Tab(28); Hex(propFlags); _

Tab(36); "LCStream"; Tab(48); propStrm.text

```

Case LCTYPE_CURRENCY:
    Set propCurr = connect.GetPropertyCurrency (tokenId)
    Print propName; Tab(20); Hex(tokenId); Tab(28); Hex(propFlags); _
    Tab(36); "LCCurrency"; Tab(48); propCurr.text
Case LCTYPE_FLOAT:
    propFloat = connect.GetPropertyFloat (tokenId)
    Print propName; Tab(20); Hex(tokenId); Tab(28); Hex(propFlags); _
    Tab(36); "Double"; Tab(48); Cstr(propFloat)
Case LCTYPE_INT:
    If (propFlags And LCPROPERTYF_BOOLEAN) Then
        propBool = connect.GetPropertyBoolean (tokenId, False)
        Print propName; Tab(20); Hex(tokenId); Tab(28); Hex(propFlags); _
        Tab(36); "Boolean"; Tab(48); Cstr(propBool)
    Else
        propInt = connect.GetPropertyInt (tokenId)
        Print propName; Tab(20); Hex(tokenId); Tab(28); Hex(propFlags); _
        Tab(36); "Long"; Tab(48); Cstr(propInt)
    End If
End Select
Loop _
While connect.ListProperty (LCLIST_NEXT, _
tokenId, propType, propFlags, propName)
End Sub

```

Example Output

NAME	ID	FLAGS	TYPE	VALUE
-----	-----	-----	-----	-----
Name	30004	4	LCStream	oracle
IsConnected	3000C	6	Boolean	False
Server	10001	1	LCStream	mycyclone
Userid	10003	1	LCStream	scott
Metadata	10005	0	LCStream	
Index	10006	0	LCStream	
MapByName	10007	2	Boolean	False
Writeback	10008	2	Boolean	False
Condition	1000B	0	LCStream	
StampField	1000C	0	LCStream	
BaseStamp	1000D	0	LCDatetime	
MaxStamp	1000E	0	LCDatetime	
TextFormat	1000F	4	Long	65535
CharacterSet	30008	4	LCStream	NATIVE
Procedure	10010	0	LCStream	
Owner	10011	0	LCStream	
AlternateMetadata	10013	2	Boolean	False
CommitFrequency	1	0	Long	0
RollbackOnError	2	2	Boolean	False
CreateLongColumn	3	0	LCStream	
CreateLongByUser	4	0	Long	0
TraceSQL	5	2	Boolean	False

Insert method for LCConnection

This method inserts a specified number of records into the connection metadata.

Note that you can achieve optimal performance by using the same fieldlist across consecutive inserts from the same target.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Insert**(*srcFieldlist*, *recordIndex*, *recordCount*)

Parameters

<i>srcFieldlist</i>	LCFieldlist. Fieldlist containing the records to insert. For each field in Fieldlist without the flag LCFIELDF_NO_INSERT, data from the corresponding field in the result set will be copied into that field. Fields in the result set and fieldlist are matched by name if the MapByName property is TRUE, by position otherwise, and are type-checked before inserting data.
<i>recordIndex</i>	Long. Optional. Starting record index of the insert in the fieldlist. The default is 1.
<i>recordCount</i>	Long. Optional. Number of records to insert. The number of records actually inserted may be less than this number if an error was encountered. While all Connectors can insert multiple records, only Connectors that indicate support for Array Insert perform a true multi-record insert and therefore reduce network traffic and increase performance. The default is 1.

Return Value

count Long. Number of records successfully inserted.

Example

Option Public
Uselsx "**Isxlc"

Sub Initialize

```
Dim src As New LCConnection ("db2")
Dim fields As New LCFieldList (5)
Dim field As LCField
Dim data(4) As String
Dim idata(4) As Long
```

```
REM set the appropriate properties to connect to the data sources
src.Database = "Gold"
src.Userid = "JDoe"
src.Password = "xyzzzy"
src.Metadata = "customer"
src.MapByName = True
REM connect to the two data sources
src.Connect
```

```
REM use a key to find certain records to remove
Set field = fields.Append ("ACCOUNTMANAGER", LCTYPE_INT)
idata(0) = 100
idata(1) = 200
idata(2) = 300
idata(3) = 400
idata(4) = 200
field.value = idata
Set field = fields.Append ("CONTACTNAME", LCTYPE_TEXT)
data(0) = "Peter Pan"
data(1) = "Big Steel"
data(2) = "R. U. Happy"
data(3) = "Issac Bernard Mathews"
data(4) = "Paula Falderall"
field.value = data
Set field = fields.append ("COMPANYADDRESS", LCTYPE_TEXT)
data(0) = "One Bit Tree"
data(1) = "Gurder Way"
data(2) = "Daisy Hill Pup Farm"
data(3) = "Big Blue Ave."
data(4) = "Planet Hollywood"
field.value = data
Set field = fields.Append ("COMPANYCITY", LCTYPE_TEXT)
data(0) = "Never Never"
data(1) = "Iron"
data(2) = "Beagle"
data(3) = "New York"
data(4) = "Parthenon"
```



```
field.value = data
Set field = fields.Append ("COMPANYSTATE", LCTYPE_TEXT)
data(0) = "Land"
data(1) = "PA"
data(2) = "WI"
data(3) = "NY"
data(4) = "AQ"
field.value = data
REM we can perform a keyed delete of all matching records in the table
Print "Inserted " & Cstr (src.Insert (fields, 1, 5)) & " record(s)."
End Sub
```

Example Output

Inserted 5 record(s).

ListProperty method for LCConnection

This method obtains the first or next property information for properties supported by this Connector.

Defined In

LCConnection

Syntax

Call *lcConnection.listProperty* (*list*, *propertyToken*, *dataType*, *propertyFlags*, *propertyName*)

Parameters

<i>list</i>	Long. Constant indicating whether to return the first or next Connector property. LCLIST_FIRST Return the first property in the property list. LCLIST_NEXT Return the next property (or the first property if this is the first call to this function for this Connection).
<i>propertyToken</i>	Long. Optional. Token assigned to the property.
<i>dataType</i>	Long. Optional. Data type of the property.
<i>propertyFlags</i>	Long. Optional. Property flags for the property; one or more of the flags below Ored together. LCPROPERTYF_CONNECT Property is used for connecting LCPROPERTYF_BOOLEAN Property is a boolean value LCPROPERTYF_READONLY Property is read-only

LCPROPERTYF_TEXTLIST

Property is a text list

PropertyName

String. Optional. Name of the property.

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
    Dim connect As New LCConnection ("oracle")
    Dim conFld As LCField
    Dim propName As String
    Dim tokenId As Long
    Dim propType As Long
    Dim propFlags As Long

    ' set some connector properties
    connect.Server = "Rainbow"
    connect.Userid = "JDoe"
    connect.Password = "xyzyzy"
    connect.Metadata = "scott.bigtable"
    connect.FieldNames = "name, address, city, state, zipcode, phone"

    Print "NAME" Tab(20); "ID"; Tab(26); "FLAGS"; _
    Tab(32); "TYPE"; Tab(38); "VALUE"
    Print "-----" Tab(20); "----"; Tab(26); "----"; _
    Tab(32); "----"; Tab(38); "-----"

    ' all of the parameters are optional and any may be omitted
    Call connect.ListProperty (LCLIST_FIRST, _
    tokenId, propType, propFlags, propName)
    Do
        Set conFld = connect.GetProperty (tokenId)
        Print propName; Tab(20); Hex(tokenId); Tab(27); Hex(propFlags); _
        Tab(32); propType; Tab(38); conFld.Text(0)
        Loop While connect.ListProperty (LCLIST_NEXT, _
        tokenId, propType, propFlags, propName)
End Sub
```

Example Output

NAME	ID	FLAGS	TYPE	VALUE
-----	-----	-----	-----	-----
Name	30004	4	6	oracle
IsConnected	3000C	6	1	0
Server	10001	1	6	mycyclone
Userid	10003	1	6	scott
Password	10004	1	7	
Metadata	10005	0	6	scott.bigtable
Index	10006	0	6	
MapByName	10007	2	1	0
Writeback	10008	2	1	0
OrderNames	1000A	8	7	
FieldNames	10009	8	7	name, address, city, state,
zipcode, phone				
Condition	1000B	0	6	
StampField	1000C	0	6	
BaseStamp	1000D	0	5	
MaxStamp	1000E	0	5	
TextFormat	1000F	4	1	65535
CharacterSet	30008	4	6	NATIVE
Procedure	10010	0	6	
Owner	10011	0	6	
AlternateMetadata	10013	2	1	0
CommitFrequency	1	0	1	0
RollbackOnError	2	2	1	0
CreateLongColumn	3	0	6	
CreateLongByUser	4	0	1	0
TracesQL	5	2	1	0

LookupProperty method for LCConnection

This method determines if a Connector supports a specified property.

Defined In

LCConnection

Syntax

flag = *lcConnection.LookupProperty(propertyToken, dataType, propertyFlags, propertyName)*

Parameters

<i>propertyToken</i>	A token identifying the Connection property. See Appendix B for a list of property tokens.
<i>dataType</i>	Long. Optional. The property data type.
<i>propertyFlags</i>	Long. Optional. The property flags from the following list. LCPROPERTYF_CONNECT Property is used for connecting. LCPROPERTYF_BOOLEAN Property is a boolean value. LCPROPERTYF_READONLY Property is read-only. LCPROPERTYF_TEXTLIST Property is a text list.
<i>propertyName</i>	String. Optional. The name of the property.
Return Value	
<i>flag</i>	TRUE or FALSE, indicating whether the property is supported for this connection.

LookupProperty method for LCCConnection

Example

Option Public
Uselsx "**Isxlc"

Sub Initialize

Dim connect As New LCCConnection ("oracle")

REM note that a connect is not necessary to list and lookup properties

REM the parameters are optional, to test for the existence of a property

REM simply provide the TOKEN parameter

If (connect.LookupProperty (LCTOKEN_SERVER)) Then

Print "Oracle has a 'Server' property"

Else

Print "Oracle does not have a 'Server' property"

End If

If (connect.LookupProperty (LCTOKEN_DATABASE)) Then

Print "Oracle has a 'Database' property"

Else

Print "Oracle does not have a 'Database' property"

End If

End Sub

Example Output

Oracle has a 'Server' property

Oracle does not have a 'Database' property

Remove method for LCConnection

This method performs either a writeback result set remove or a keyed remove of records in the external system.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Remove**(*keyFieldlist*, *recordIndex*, *recordCount*)

Parameters

<i>keyFieldlist</i>	LCFieldlist. Fieldlist containing keys for a keyed remove only. For each field in the Fieldlist with the flag LCFIELDF_KEY set, only records in the connection with the same value for that field are deleted. Use LCFIELDF_KEY_XXX flags as inequality keys, if desired. Zero or more key fields may be supplied, with zero keys resulting in all records in the target being deleted. Fields in the connection and fieldlist are matched by name if the MapByName property is TRUE, by position otherwise.
<i>recordIndex</i>	Long. Optional. Starting record index in the fieldlist. The default is 1.
<i>recordCount</i>	Long. Optional. Number of keyed remove operations to perform. <i>RecordCount</i> must be 1 for a writeback remove but may be greater to perform multiple keyed updates with different update and key values. The default is 1.

Return Value

<i>count</i>	Long. Number of records successfully removed. If this number cannot be determined, the constant LCCOUNT_UNKNOWN is returned.
--------------	--

Usage Notes

You can achieve optimal performance by using the same fieldlist across consecutive removes from the same target.

The property Writeback indicates whether to perform a writeback or keyed remove:

- **Writeback remove:** If the Writeback property is now set, this method removes the most recently fetched record from the writeback result set (the result set produced by LCConnection.Execute, LCConnection.Select or LCConnection.Call with the Writeback property set).
- **Keyed remove:** Removes all records in the supplied metadata with field values matching all fields in the fieldlist that have the LCFIELD_KEY field flag set. If a value is specified for the property Condition, this will be included in the key search criteria. This value must be in a syntax valid for the relevant Connector. See the documentation for the Connector for information about its supported syntax.

When using inequality key flags GT, LT, and NE, it is important to remember that the default of no flags is equal. The following combinations are valid for inequality flags:

- equal to LCFIELD_KEY
- greater than or equal to LCFIELD_KEY + LCFIELD_KEY_GT
- less than or equal to LCFIELD_KEY + LCFIELD_KEY_LT
- not equal to LCFIELD_KEY + LCFIELD_KEY_NE
- greater than LCFIELD_KEY + LCFIELD_KEY_GT + LCFIELD_KEY_NE
- less than LCFIELD_KEY + LCFIELD_KEY_LT + LCFIELD_KEY_NE

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
  Dim src As New LCConnection ("db2")
  Dim keyList As New LCFieldList
  Dim key As LCField
```

```
  REM set the appropriate properties to connect to the data sources
```

```
  src.Database = "Gold"
```

```
  src.Userid = "JDoe"
```

```
  src.Password = "xyzyzy"
```

```
  src.Metadata = "customer"
```

```
  src.MapByName = True
```

```
  REM connect to the two data sources
```

```
  src.Connect
```

```
  REM use a key to find certain records to remove
```

```
  Set key = keyList.Append ("ACCOUNTMANAGER", LCTYPE_INT)
```

```
  key.Flags = LCFIELD_KEY
```

```
  key.value = 200
```

```
  REM we can perform a keyed delete of the matching record in the table
```

```
  Print "Removed " & Cstr (src.Remove (keyList)) & " record(s)"
```

```
End Sub
```

Example Output

```
Removed 2 record(s)
```

Select method for LCConnection

This method produces a result set from the current METADATA property and other properties.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Select** (*keyFieldlist*, *recordIndex*, *destFieldlist*)

Parameters

<i>keyFieldlist</i>	LCFieldlist. Selection keys. All fields in <i>KeyFieldlist</i> with the LCFIELDF_KEY flag set are used as the selection keys. Only records in the connection with the same value for key fields will be selected. Additional LCFIELDF_KEY_XXX flags (GT - greater than, LT - less than, NE - not equal) allow inequality keys to be used. Zero or more key fields may be supplied to restrict the result set. No keys or Nothing will select all records for the result set.
<i>recordIndex</i>	Long. Record index position in the key fieldlist from which to obtain the record containing key field values.
<i>destFieldlist</i>	LCFieldlist. Fieldlist to contain the metadata of the selected result set. The fields in the result set will be appended to this fieldlist. If the result set metadata is not required, use Nothing.

Return Value

<i>count</i>	Long. Number of records in the result set. If this number cannot be determined by the Connector, the constant LCCOUNT_UNKNOWN is returned.
--------------	--

Usage Notes

The property Writeback indicates whether to perform a writeback or keyed selection:

- **Writeback selection:** If the Writeback property is now set, this method selects fields in the most recently fetched record from the writeback result set (the result set produced by LCConnection.Execute, LCConnection.Select or LCConnection.Call with the Writeback property set).
- **Keyed selection:** Selects all records in the supplied metadata with field values matching all fields in the fieldlist that have the LCFIELD_KEY field flag set. If a value is specified for the property Condition, this will be included in the key search criteria. This value must be in a syntax valid for the relevant Connector. See the documentation for the Connector for information about its supported syntax.

When using inequality key flags GT, LT, and NE, it is important to remember that the default of no flags is equal. The following combinations are valid for inequality flags:

- equal to LCFIELD_KEY
- greater than or equal to LCFIELD_KEY + LCFIELD_KEY_GT
- less than or equal to LCFIELD_KEY + LCFIELD_KEY_LT
- not equal to LCFIELD_KEY + LCFIELD_KEY_NE
- greater than LCFIELD_KEY + LCFIELD_KEY_GT + LCFIELD_KEY_NE
- less than LCFIELD_KEY + LCFIELD_KEY_LT + LCFIELD_KEY_NE

Select method for LCConnection

Example

```
Option Public  
Uselsx "*Isxlc"
```

```
Sub Initialize
```

```
    Dim src As New LCConnection ("db2")  
    Dim fldLst As New LCFieldList  
    Dim fld As LCField  
    Dim count As Integer
```

```
    REM set the appropriate properties to connect to the data sources
```

```
    src.Database = "Gold"  
    src.Userid = "JDoe"  
    src.Password = "xyzyz"  
    src.Metadata = "customer"
```

```
    REM connect to the two data sources
```

```
    src.Connect
```

```
    REM now connected, we can execute a selection statement
```

```
    count = src.Select (Nothing, 1, fldLst)
```

```
    Select Case count
```

```
    Case LCCOUNT_UNKNOWN
```

```
        Print "An unknown number of records were located."
```

```
    Case 0
```

```
        Print "No data were located."
```

```
    Case Else
```

```
        Print "The table contains " & Cstr(count) & " records."
```

```
    End Select
```

```
End Sub
```

Example Output

An unknown number of records were located.

SetProperty method for LCConnection

This method assigns the value for a connection property. Note that use of dynamic properties is more efficient.

Defined In

LCConnection

Syntax

Call *thisConnection.SetProperty(propertyToken, srcField)*

Parameters

<i>propertyToken</i>	Long. Token representing the Connector property for which the value is to be assigned. See Appendix B for a list of tokens.
<i>srcField</i>	LCField. New value for the Connector property. If the property value is of a different type than this field, then data conversion will occur. The property value will be assigned from the first value in the field.

SetProperty method for LCConnection

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize
```

```
    Dim connect As New LCConnection ("oracle")
```

```
    Dim conFld As New LCField (LCTYPE_TEXT, 1)
```

```
    REM set appropriate properties to connect to oracle
```

```
    conFld.text = "Barker"
```

```
    Call connect.SetProperty (LCTOKEN_SERVER, conFld)
```

```
    REM example of using the expanded property
```

```
    Print "Here is the server property value: " & connect.Server
```

```
End Sub
```

Example Output

Here is the server property value: Barker

SetProperty<Type> methods for LCConnection

These methods assign a Connection property value as a specific data type. Datatypes supported include: Boolean, Currency, Datetime, Float, Int, Numeric, and Text.

Defined In

LCConnection

Syntax

Call *lcConnection.SetPropertyBoolean(propertyToken, srcBoolean)*

Call *lcConnection.SetPropertyCurrency(propertyToken, srcCurrency)*

Call *lcConnection.SetPropertyDatetime(propertyToken, srcDatetime)*

Call *lcConnection.SetPropertyFloat(propertyToken, srcFloat)*

Call *lcConnection.SetPropertyInt(propertyToken, srcInt)*

Call *lcConnection.SetPropertyNumeric(propertyToken, srcNumeric)*

Call *lcConnection.SetPropertyStream(propertyToken, srcStream)*

Parameters

<i>propertyToken</i>	Long. Token identifying the Connection property. See Appendix B for a list of tokens.
<i>src<Type></i>	Value to assign to the Connection property.

SetProperty<Type> Examples

Option Public

Uselsx "**Isxlc"

Sub Initialize

```
Dim connect As New LCConnection ("db2")
Dim conFld As New LCField (LCTYPE_TEXT,1)
Dim propName As String
Dim propDate As New LCDateTime
Dim propNumeric As New LCNumeric
Dim propStrm As New LCStream (0, 0, LCTYPE_TEXT)
Dim propCurr As New LCCurrency
Dim propFloat As Double
Dim propInt As Long
Dim tokenId As Long
Dim propType As Long
Dim propFlags As Long
```

```
' set the appropriate properties to connect to DB2,
' note that you can also use dynamic properties to do this
connect.Database = "Gold"
connect.Userid = "JDoe"
connect.Password = "xyzzzy"
```

```
Call connect.ListProperty (LCLIST_FIRST,_
tokenId, propType, propFlags, propName)
```

Do

```
Set conFld = connect.GetProperty (tokenId)
If (propFlags And LCPROPERTYF_READONLY) <> LCPROPERTYF_READONLY Then
' match the property to a datatype and set it
Select Case propType
Case LCTYPE_DATETIME:
propDate.SetCurrent
Call connect.SetPropertyDatetime (tokenId, propDate)
Print propName & " is now a datetime and contains " & propDate.text
Case LCTYPE_NUMERIC:
propNumeric.text = "100.0123456789"
Call connect.SetPropertyNumeric (tokenId, propNumeric)
Print propName & " is now a numeric and contains " & propNumeric.text
Case LCTYPE_TEXT:
propStrm.text = "a beautiful day"
Call connect.SetPropertyStream (tokenId, propStrm)
Print propName & " is now text and contains " & propStrm.text
Case LCTYPE_CURRENCY:
propCurr.text = "140.10"
Call connect.SetPropertyCurrency (tokenId, propCurr)
Print propName & " is now currency and contains " & propCurr.text
Case LCTYPE_FLOAT:
```


SetProperty<Type> methods for LCConnection

```
Call connect.SetPropertyFloat (tokenId, 30000.456)
Print propName & " is now a float and contains " & Cstr (30000.456)
Case LCTYPE_INT:
If (propFlags And LCPROPERTYF_BOOLEAN) Then
    Call connect.SetPropertyBoolean (tokenId, True)
    Print propName & " is now an int and contains " & Cstr(True)
Else
    Call connect.SetPropertyInt (tokenId, 123)
    Print propName & " is now an int and contains " & Cstr(123)
End If
End Select
End If
Loop _
While connect.ListProperty (LCLIST_NEXT, _
tokenId, propType, propFlags, propName)
End Sub
```

Example Ouput

Database is now text and contains a beautiful day

```
Userid is now text and contains a beautiful day
Metadata is now text and contains a beautiful day
Index is now text and contains a beautiful day
MapByName is now an int and contains True
Writeback is now an int and contains True
Condition is now text and contains a beautiful day
StampField is now text and contains a beautiful day
BaseStamp is now a datetime and contains 09/08/1998 05:24:33.65 PM
MaxStamp is now a datetime and contains 09/08/1998 05:24:33.65 PM
Procedure is now text and contains a beautiful day
Owner is now text and contains a beautiful day
AlternateMetadata is now an int and contains True
CommitFrequency is now an int and contains 123
RollbackOnError is now an int and contains True
CreateMaxLogged is now an int and contains 123
NoJournal is now an int and contains True
CreateInDatabase is now text and contains a beautiful day
TraceSQL is now an int and contains True
```

Update method for LCConnection

This method updates selected records in the connection metadata.

Defined In

LCConnection

Syntax

```
count = lcConnection.Update(srcFieldlist, recordIndex, recordCount)
```

Parameters

<i>srcFieldlist</i>	LCFieldlist. The fieldlist that contains the fields to be changed.
<i>recordIndex</i>	Long. Optional. The starting record in the fieldlist. The default is 1.
<i>recordCount</i>	Long. Optional. The number of records in the fieldlist to use to perform the update. The default is 1.

Return Value

<i>count</i>	Long. Number of records successfully updated. This may be LCCOUNT_UNKNOWN.
--------------	--

Usage Notes

You can achieve optimal performance by using the same fieldlist across consecutive updates to the same target.

The property Writeback indicates whether to perform a writeback or keyed update:

- Writeback update: If the Writeback property is now set, this method updates fields in the most recently fetched record from the writeback result set (the result set produced by LCConnection.Execute, LCConnection.Select or LCConnection.Call with the Writeback property set). Fields that have the NO_UPDATE field flag set are not updated.

- **Keyed update:** Updates all records in the supplied metadata with field values matching all fields in the fieldlist that have the LCFIELDF_KEY field flag set. Fields with the NO_UPDATE field flags set are not affected. If a value is specified for the property Condition, this will be included in the key search criteria. This value must be in a syntax valid for the relevant Connector. See the documentation for the Connector for information about its supported syntax.

When using inequality key flags GT, LT, and NE, it is important to remember that the default of no flags is equal. The following combinations are valid for inequality flags:

- equal to LCFIELDF_KEY
- greater than or equal to LCFIELDF_KEY + LCFIELDF_KEY_GT
- less than or equal to LCFIELDF_KEY + LCFIELDF_KEY_LT
- not equal to LCFIELDF_KEY + LCFIELDF_KEY_NE
- greater than LCFIELDF_KEY + LCFIELDF_KEY_GT + LCFIELDF_KEY_NE
- less than LCFIELDF_KEY + LCFIELDF_KEY_LT + LCFIELDF_KEY_NE

Update method for LCConnection

Example

```
Option Public
Option Explicit
Uselsx "**!sxlc"
```

```
Sub Initialize
```

```
    Dim src As New LCConnection ("db2")
```

```
    Dim fldList As New LCFieldList
```

```
    Dim fld As LCField
```

```
    ' set the appropriate properties to connect to the data source
```

```
    src.Database = "Gold"
```

```
    src.Userid = "JDoe"
```

```
    src.Password = "xyzyz"
```

```
    src.Metadata = "customer"
```

```
    src.Connect
```

```
    ' use a key to find certain records to update
```

```
    Set fld = fldList.Append ("ACCOUNTMANAGER", LCTYPE_INT)
```

```
    fld.Flags = LCFIELD_KEY
```

```
    fld.value = 200
```

```
    ' set the field which will be changed, and set the new value
```

```
    Set fld = fldList.Append ("CONTACTNAME", LCTYPE_TEXT)
```

```
    fld.text = "Me"
```

```
    src.MapbyName = True
```

```
    ' set the contact's city to "Denver"
```

```
    ' for the record who's contact is "Me"
```

```
    Print "The update affected " & Cstr (src.Update (fldList)) & " records"
```

```
End Sub
```

Example Output

The update affected 2 records

Chapter 3

LCCurrency Class

This chapter provides information about the Lotus Connectors LCCurrency class methods and properties.

Overview

The LCCurrency class represents a currency value and has the same format and restrictions as the LotusScript currency datatype. The value is an 8-byte integer with a fixed scale of 4 decimal places, providing 19 digits of precision. Currency is commonly used when higher precision is required, such as for monetary amounts. A currency is much more precise than an integer, more precise than a float, and more efficient (but less precise) than a numeric. Note that during any currency overflow, the maximum or minimum valid currency value is assigned in addition to the error generated.

Type CURRENCY format and values

Type constant	LCTYPE_CURRENCY
Description	8-byte integer with a fixed scale of 4
Other	Precision (LCMAX_CURRENCY_PREC) = 19
	Scale (LCMAX_CURRENCY_SCALE) = 4
	Minimum Value (LCMIN_CURRENCY_VALUE) = -922,337,203,685,477.5807
	Maximum Value (LCMAX_CURRENCY_VALUE) = 922,337,203,685,477.5807

LCCurrency Class Methods Summary

The following are the LCCurrency class methods:

LCCurrency.Add	Adds two currency values and deposits the result in the object making the call.
LCCurrency.Compare	Compares two currency values returning a value indicating the relationship between them.
LCCurrency.Copy	Makes a copy of a currency.
LCCurrency.Subtract	Subtracts one currency value from another, and returns the result to the object making the call.

LCCurrency Properties

LCCurrency has two properties: Text and Value.

- Text Text is a string representation.
- Value Value is of LotusScript currency data type.

New method for LCCurrency

This is the constructor for LCCurrency. It initializes a currency object.

Defined In

LCCurrency

Syntax

Dim *variableName* as **New** LCCurrency

Add method for LCCurrency

Adds two LCCurrency values, producing the sum, which is returned in the object making the call.

Defined In

LCCurrency

Syntax

Call *currencyTotal*.**Add**(*currency1*, *currency2*)

Parameters

currency1 LCCurrency. The first of the two LCCurrencies to add.

currency2 LCCurrency. The second of the two LCCurrencies to add.

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim num1 As New LCCurrency  
    Dim num2 As New LCCurrency  
    Dim sum As New LCCurrency  
  
    num1.Value = 12345.6789  
    num2.Value = 12345.6789  
    Call sum.Add (Num1, Num2)  
    Print "The sum of the two currencies is " & sum.Text  
End Sub
```

Example Output

The sum of the two currencies is 24691.3578

Compare method for LCCurrency

Compares two currency values and returns the relationship.

Defined In

LCCurrency

Syntax

Result = *thisCurrency*.**Compare**(*baseCurrency*)

Parameters

<i>baseCurrency</i>	LCCurrency. The base currency value to which to compare <i>thisCurrency</i> .
---------------------	---

Return Value

<i>Result</i>	Result of the comparison: Result > 0 (positive): <i>thisCurrency</i> is greater than <i>baseCurrency</i> . Result < 0 (negative): <i>thisCurrency</i> is less than <i>baseCurrency</i> . Result = 0: <i>thisCurrency</i> is equal to <i>baseCurrency</i> .
---------------	---

Example

```
Option Public
Uselsx "**lsxl"

Sub Initialize
    Dim num1 As New LCCurrency
    Dim num2 As New LCCurrency

    num1.Value = 123.456789
    num2.Value = 123

    If (num1.Compare (num2) = 0) Then
        Print "The first number is the same as the second."
    ElseIf (num1.Compare (num2) > 0) Then
        Print "The first number is greater than the second."
    Else
        Print "The first number is less than the second."
    End If
End Sub
```

Example Output

The first number is greater than the second.

Copy method for LCCurrency

This method makes a copy of an LCCurrency object.

Defined In

LCCurrency

Syntax

Set *newCurrency* = *srcCurrency*.**Copy**

Parameters

<i>srcCurrency</i>	LCCurrency. The source currency value to be copied.
--------------------	---

Return Value

<i>newCurrency</i>	LCCurrency. The copy of the <i>srcCurrency</i> object.
--------------------	--

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize  
    Dim num1 As New LCCurrency  
    Dim num2 As LCCurrency  
  
    num1.Value = 12345.6789  
    Set num2 = num1.Copy  
    Print "The copy has a value of " & num2.Text  
End Sub
```

Example Output

The copy has a value of 12345.6789

Subtract method for LCCurrency

This method subtracts one LCCurrency value from another, producing the result.

Defined In

LCCurrency

Syntax

Call *currencyResult*.**Subtract**(*currency1*, *currency2*)

Parameters

<i>currency1</i>	LCCurrency. This is the initial LCCurrency from which you want to subtract the second currency.
<i>currency2</i>	LCCurrency. The LCCurrency to be subtracted from <i>currency1</i> .

Example

Option Public

Uselsx "**lsxlc"

Sub Initialize

Dim num1 As New LCCurrency

Dim num2 As New LCCurrency

Dim diff As New LCCurrency

num1.Value = 98765.4321

num2.Value = 12345.6789

Call diff.Subtract (Num1, Num2)

Print "The difference of the two currencies is " & diff.Text

End Sub

Example Output

The difference of the two currencies is 86419.7532

Chapter 4

LCDatetime Class

This chapter provides information about the Lotus Connectors LCLDatetime class methods and properties.

Overview

The LCLDatetime class represents a specific date and time, including timezone and daylight savings time information. A datetime value may have either its date or time component unavailable, indicated by special constant values for date (LCLDTNULL_DATE) or time (LCLDTNULL_TIME) components. Flags control the behavior for the existence/absence of the date or time portion. During any datetime overflow, the maximum or minimum valid datetime value is assigned in addition to the error return.

The time portion of the datetime is precise to hundredths. Some data sources do not support this precision, or support greater precision. For these systems it may be necessary to set the field flags for all datetime datatype fields to LCFIELDF_TRUNC_PREC, to avoid a precision loss error.

Type DATETIME format and values

Type constant	LCTYPE_DATETIME
Description	Date and time value with time zone and Daylight Savings Time (DST)
Other	Precision = 0.01 second
	Minimum Year Value (LCMIN_DATETIME_YEAR) = 1
	Maximum Year Value (LCMAX_DATETIME_YEAR) = 32767

LCDatetime Class Methods Summary

LCDatetime.Adjust	Alters a Datetime by a specified number of units.
LCDatetime.Clear	Clears a Datetime value.
LCDatetime.Compare	Compares two Datetime values returning a value indicating the relationship between them.
LCDatetime.Copy	Makes a copy of an LCDatetime object.
LCDatetime.GetDiff	Returns the difference between two Datetimes in requested time units.
LCDatetime.SetConstant	Produces a special constant Datetime commonly used for comparisons.
LCDatetime.SetCurrent	Sets a Datetime value to the current system time.

LCDatetime Properties

LCDatetime.Minute	Long. Minute (0-59).
LCDatetime.Second	Long. Second (0-59).
LCDatetime.Hundredth	Long. Hundredths of second (0-99).
LCDatetime.Day	Long. Day of month (1-31).
LCDatetime.Hour	Long. Hour (0-23).
LCDatetime.Month	Long. Month (1-12).
LCDatetime.Year	Long. Integer indicating the year (1-32767).
LCDatetime.Weekday	Long. Integer indicating the day of the week (1-7, Sunday = 1). Output only.
LCDatetime.Zone	Long. Integer indicating the time zone (-12 to 12).
LCDatetime.DST	Boolean. Indicates whether Daylight Savings Time is in effect.
LCDatetime.Ticks	Long. Tick count representing hundredths of a second since midnight.
LCDatetime.Text	String representation.
LCDatetime.Julian	Long. Julian date.
LCDatetime.Value	LotusScript Variant containing a date/time.

New Constructor method for LCDatetime

This is the constructor for LCDatetime. It initializes a new LCDatetime object.

Defined In

LCDatetime

Syntax

Dim *variableName* as **New** LCDatetime(*Year, Month, Day, Hour, Minute, Second, Hundredth, Zone, DST*)

Parameters

All parameters are optional.

<i>Year</i>	As Long. Value in the range 1-32767. Default is 0.
<i>Month</i>	As Long. A value in the range 1-12. Default is 0.
<i>Day</i>	As Long. A value in the range 1-31. Default is 0.
<i>Hour</i>	As Long. A value in the range 0-23. Default is 0.
<i>Minute</i>	As Long. A value in the range 0-59. Default is 0.
<i>Second</i>	As Long. A value in the range 0-59. Default is 0.
<i>Hundredth</i>	As Long. A value in the range 0-99. Default is 0.
<i>Zone</i>	As Long. A value representing the time zone in the range -11-11. Default is GMT.
<i>DST</i>	As Variant. Boolean. Daylight savings time is in effect when true. Default is 0 (GMT).

Adjust method for LCDatetime

Alters a datetime by specified datetime units.

Defined In

LCDatetime

Syntax

Call *changeDatetime*.**Adjust**(*Units*, *Amount*)

Parameters

Units

Long. The unit to adjust. Units are in the order listed in the constructor: *Year*, *Month*, *Day*, *Hour*, *Minute*, *Second*, *Hundredth*, *Zone*, *DST*. Use the following constants:

LCDTUNIT_YEAR – Year units.

LCDTUNIT_MONTH - Month units.

LCDTUNIT_DAY - Day units.

LCDTUNIT_WEEKDAY - Weekday units.

LCDTUNIT_HOUR - Hour units.

LCDTUNIT_MINUTE - Minute units.

LCDTUNIT_SECOND - Second units.

LCDTUNIT_HUNDREDTH - Hundredth of second units.

LCDTUNIT_ZONE – Time zone units.

Amount

Long. The amount to adjust the datetime unit. Positive means later in time; negative means earlier.

Adjust method for LCDatetime

Example

```
Option Public  
Uselsx "*Isxlc"
```

```
Sub Initialize  
    Dim clock As New LCDatetime  
    clock.SetCurrent  
    Print "The time is " & clock.Text  
    Call clock.Adjust (LCDTUNIT_HOUR, -100)  
    Print "100 hours ago, the time was " & clock.Text  
End Sub
```

Example Output

```
The time is 09/08/1998 05:22:07.18 PM  
100 hours ago, the time was 09/05/1998 02:37:52.82 AM
```

Clear method for LCDatetime

Clears a previously set Datetime value to NULL.

Defined In

LCDatetime

Syntax

Call *dateTimeObject*.Clear

Example

```
Option Public
Uselsx "**lsxlc"

Sub Initialize
  Dim Clock As New LCDatetime (1999, 12, 31, 23, 59, 59, 99)
  Clock.Clear
  If (Clock.Text = "") Then
    Print "The time has been cleared."
  Else
    Print "The time is " & Clock.Text
  End If
End Sub
```

Example Output

The time has been cleared.

Compare method for LCDatetime

Compares two Datetime values and returns the relationship.

Defined In

LCDatetime

Syntax

Result = *thisDatetime*.**Compare**(*baseDatetime*)

Parameters

baseDatetime The Datetime to which to compare *thisDatetime*.

Return Value

Result Result of the comparison:

Result > 0 (positive): *thisDatetime* is greater than *baseDatetime*.

Result < 0 (negative): *thisDatetime* is less than *baseDatetime*.

Result = 0: *thisDatetime* is equal to *baseDatetime*.

Example

```
Option Public
Uselsx "**lsxc"

Sub Initialize
    Dim Clock As New LCDatetime (1999, 12, 31, 23, 59, 59, 99)
    Dim Match As New LCDatetime

    Match.SetCurrent

    If (Clock.Compare (Match) = 0) Then
        Print "The current time matches " & Clock.Text
    ElseIf (Clock.Compare (Match) > 0) Then
        Print "The current time is before " & Clock.Text
    Else
        Print "The current time is after " & Clock.Text
    End If
End Sub
```

Example Output

The current time is before 12/31/1999 11:59:59.99 PM

Copy method for LCDatetime

This method makes a copy of an LCDatetime object.

Defined In

LCDatetime

Syntax

Set *newDatetime* = *srcDatetime*.**Copy**

Parameters

<i>SrcDatetime</i>	LCDatetime. The datetime object that you want to copy.
--------------------	--

Return Value

<i>newDatetime</i>	LCDatetime. The copy of the srcDatetime object.
--------------------	---

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim session As New LCSession
    Dim StopTime As New LCDatetime
    Dim StartTime As LCDatetime
```

```
    StopTime.SetCurrent
    Set StartTime = StopTime.Copy
    session.Sleep (100) ' 100 milliseconds
    StopTime.SetCurrent
```

```
    Print "The difference between start and stop is:"
    Print StopTime.GetDiff (StartTime, LCDTUNIT_HUNDREDTH) & _
        " hundredths of a second."
```

```
End Sub
```

Example Output

The difference between start and stop is:
10 hundredths of a second.

GetDiff method for LCDatetime

The GetDiff method gets the difference between the values of two LCDatetime objects.

Defined In

LCDatetime

Syntax

difference = *lcDatetime*.**GetDiff**(*baseDatetime*, *Units*)

Parameters

baseDatetime LCDatetime. The datetime object to which you want to compare the *lcDatetime* object.

Units Long. The units to compare. Units are in the order listed in the constructor: *Year*, *Month*, *Day*, *Hour*, *Minute*, *Second*, *Hundredth*, *Zone*, *DST*. Use the following constants:

LCDTUNIT_YEAR – Year units.

LCDTUNIT_MONTH - Month units.

LCDTUNIT_DAY - Day units.

LCDTUNIT_WEEKDAY - Weekday units.

LCDTUNIT_HOUR - Hour units.

LCDTUNIT_MINUTE - Minute units.

LCDTUNIT_SECOND - Second units.

LCDTUNIT_HUNDREDTH - Hundredth of second units.

LCDTUNIT_ZONE – Time zone units.

Return Value-

difference The difference between the two LCDatetime objects, in the units specified.

Example

```
Option Public  
Uselx "Isxlc"
```

```
Sub Initialize
```

```
    Dim Boston As New LCDatetime _  
        (1998, 1, 25, 08, 50, 00, 00, -5, True)
```

```
    Dim Singapore As New LCDatetime _  
        (1998, 1, 27, 1, 10, 00, 00, 8, False)
```

```
    Print "A flight taking off from Boston at 8:50am"
```

```
    Print "and landing in Singapore at 1:10am"
```

```
    Print "will take " & Singapore.GetDiff (Boston, LCDTUNIT_HOUR) & " hours."
```

```
End Sub
```

Example Output

A flight taking off from Boston at 8:50am
and landing in Singapore at 1:10am
will take 27 hours.

SetConstant method for LCDatetime

Produces a special constant Datetime object.

Defined In

LCDatetime

Syntax

Call *lcDatetime.SetConstant(datetimeConstant)*

Parameters

datetimeConstant

One of the following Datetime Constants:

LCDTCONST_MINIMUM: A datetime which is less than any other datetime (except another minimum).

LCDTCONST_MAXIMUM: A datetime that is greater than any other datetime (except another maximum).

LCDTCONST_WILDCARD: A datetime that matches any other datetime. The date value is LCDTNULL_DATE and the time value is LCDTNULL_TIME.

Example

```
Option Public  
Uselsx "**Isxlc"
```

```
Sub Initialize
```

```
    Dim Clock As New LCDatetime (1999, 12, 31, 23, 59, 59, 99)
```

```
    Dim Match As New LCDatetime
```

```
    Call Match.SetConstant (LCDTCONST_WILDCARD)
```

```
    If (Clock.Compare (Match) = 0) Then
```

```
        Print "The current time matches the wild card constant."
```

```
    Else
```

```
        Print "The current time does not match the wild card constant."
```

```
    End If
```

```
End Sub
```

Example Output

The current time matches the wild card constant.

SetCurrent method for LCDatetime

This method sets the Datetime value to the current system datetime.

Defined In

LCDatetime

Syntax

Call *lcDatetimeObject*.**SetCurrent**

Example

```
Option Public  
Uselsx "**lsxc"
```

```
Sub Initialize  
    Dim stopWatch As New LCDatetime  
    stopWatch.setcurrent  
    Print "The time is " & stopWatch.text  
End Sub
```

Example Output

The time is 09/08/1998 05:22:02.85 PM

Chapter 5

LCField Class

This chapter provides information about the LCField class methods and properties.

Overview

The LCField class represents a data object containing one or more data values of a designated data type. A field may be an independent repository for data or may be a reference to another field, as in when getting a field from a fieldlist. In this case, changes to data in the field affect the contents of the data referenced by the fieldlist.

Note that in repetitive operations such as data fetch - insert loops, it is more efficient to get a reference field from a fieldlist prior to the loop, and act on the data through the reference than to locate the data field in each iteration of the loop.

LCField Class Methods Summary

The methods for the LCField class are summarized below.

Allocation

The following methods create and free Field instances.

New LCField	(Constructor) Allocates a new Field object instance.
LCField.Copy	Creates a new Field object instance as a copy of another field. The field data is also copied.
Delete LCField	(Destructor) Frees a Field object instance allocated with the constructor or LCField.Copy. This method does not free fields created as part of a fieldlist.

Field Properties

The following methods support retrieval of field properties and assignment of mutable field properties.

LCField.GetType	Retrieves the data type for a field.
LCField.GetValueCount	Retrieves the number of data values for a field.
LCField.GetFlags	Retrieves the current field flags for a field.
LCField.SetFlags	Assigns the current field flags for a field.
LCField.GetVirtualCode	Checks if a field has this virtual code set.
LCField.ListVirtualCode	Lists the virtual code(s) for a field.
LCField.SetVirtualCode	Sets a virtual code for a field.

Field Format

The following methods support retrieval of field format information. Setting the format for a field clears all current data values for that field.

LCField.GetFormatNumber	Retrieves the current format settings for a number field.
LCField.SetFormatNumber	Assigns the current format settings for a number field.
LCField.GetFormatDatetime	Retrieves the current format settings for a datetime field.
LCField.SetFormatDatetime	Assigns the current format settings for a datetime field.
LCField.GetFormatStream	Retrieves the current format settings for a stream field.
LCField.SetFormatStream	Assigns the current format settings for a stream field.

Field Data

The following methods support access to and modification of field data and NULL indicators.

LCField.IsNull	Queries whether a specific field data value is NULL.
LCField.SetNull	Assigns the NULL indicator for a specific field data value.

LCField.Get<Type>	Retrieves a specific field data value as a particular data type, converting the data if necessary.
LCField.Set<Type>	Assigns a specific field data value from a particular data type, converting the data if necessary.

Miscellaneous

LCField.Compare	Compare data values between two fields
LCField.Convert	Convert data values between two fields

LCField Properties

The following are the properties for the LCField class:

Count	Long. One or greater. Data space for this many data values and NULL indicators will be allocated. The value count is automatically assigned for fields in a fieldlist based on the fieldlist record count (see Fieldlist description). Assigned at creation and Read-Only.
Datatype	Long. One of the Lotus Connector datatypes. Assigned at creation and Read-Only.
Flags	Long. Zero or more of the following field flags OR-ed together. LCFIELDF_NO_NULL Field cannot be NULL LCFIELDF_TRUNC_PREC Allow precision truncation LCFIELDF_TRUNC_DATA Allow data truncation LCFIELDF_NO_FETCH Do not FETCH this field LCFIELDF_NO_INSERT Do not INSERT this field LCFIELDF_NO_UPDATE Do not UPDATE this field LCFIELDF_NO_REMOVE Do not REMOVE this field LCFIELDF_NO_CREATE Do not CREATE this field LCFIELDF_NO_DROP Do not DROP this field LCFIELDF_KEY Field is a KEY for keyed operations LCFIELDF_KEY_GT Key condition is greater than LCFIELDF_KEY_LT Key condition is less than LCFIELDF_KEY_NE Key condition is not equal to LCFIELDF_KEY_LIKE Key condition is like (native pattern match)
IsNull	Boolean. True or False.

Text *Array* of string representations.

Value *Array* of LotusScript datatypes. The value or values contained depend on the datatype of the field, as listed below.

Field Type	Value Return Type
LCTYPE_CURRENCY	Currency
LCTYPE_DATETIME	Variant (Date/Time)
LCTYPE_INT	Long
LCTYPE_FLOAT	Double
LCTYPE_NUMERIC	Double
LCTYPE_TEXT	String
LCTYPE_BINARY	String

Field Format

The format of a field is specific to its general class of type: number (int, float, currency, numeric), datetime, or stream (text and binary). For all format values, zero indicates that the indicated information is not used for this field. The information for each datatype is indicated below.

NOTE: Number and Datetime formats do not affect the actual data, but are used for data creation only. Stream format does affect the actual data.

Number (INT, FLOAT, CURRENCY, NUMERIC)

Flags: zero or more of the following constants. Multiple constants can be ORed together:

LCNUMBERF_UNSIGNED	Source type is unsigned
LCNUMBERF_NUMERIC	Source type is NUMERIC (unnecessary with LCTYPE_NUMERIC)
LCNUMBERF_DECIMAL	Source type is DECIMAL
LCNUMBERF_PACKED	Source numeric/decimal data is packed
LCNUMBERF_BIT	Source type is a bit

Size: size, in bytes, of this number value. Zero indicates to default to the size of the type for this field.

Precision: digits of precision for this value. Zero indicates not used.

Scale: numeric scale. A true scale of zero uses the constant LCSCALE_ZERO. Zero indicates not used.

Datetime (DATETIME)

Flags: zero or more of the following constants. Multiple constants can be ORed together:

LCDATETIMEF_NO_DATE	Source type is time only
LCDATETIMEF_NO_TIME	Source type is date only

Size: size, in bytes, of this datetime value. Zero indicates to default to the size of the type for this

field.

Stream (TEXT, BINARY)

Flags: zero or more of the stream flags LCSTREAMF_xxx. Multiple flags can be ORed together. See the Stream class description.

MaxLength: maximum length, in bytes, of this stream value. Zero indicates no maximum length. See the Stream class description.

StreamFormat: default stream format for this stream value. Zero indicates no default stream format. Use one of the LCSTREAMFMT_xxx constants.

Field Virtual Codes - Advanced Usage

Virtual codes allow specific fields to be interpreted differently for Connectors which support virtual fields. For example, a Connector could support a virtual field named "RecordId", which holds a special internal record indicator for that Connector. This Connector would interpret this field differently than normal data, while other Connectors would consider this field a standard data field. While the virtual code indicates special handling, the field name determines the type of handling on a Connector-specific basis.

To achieve this functionality, the virtual code for a field is set to match either a Connector code or a Connection code. A Connector code is a code which is the same for all connections to a particular Connector in a Session. The Connector code can be obtained by retrieving the property LCTOKEN_CONNECTOR_CODE with LCConnection.GetProperty or as the code returned by LCSession.ListConnector or LCSession.LookupConnector. Using a Connector code as a virtual code causes all connections to that Connector to interpret the field as a virtual field. A connection code is a code which is different for all connections. The Connect code can be obtained by retrieving the property LCTOKEN_CONNECTION_CODE with LCConnection.GetProperty. Using a Connection code as a virtual code causes only that specific Connection to interpret the field as a virtual field. The Connector code can also be determined by taking the Connection code and zeroing the low two bytes (OR with LCMASK_CONNECTOR_CODE). Note that these codes are dynamically assigned and must be obtained for each execution of a script.

New method for LCField

This is the constructor method for LCField. It initializes an LCField object.

Defined In

LCField

Syntax

Dim *variablename* as **New** LCField(*type*, *count*)

Parameters

<i>type</i>	Long. Data type of the field object, one of the following constants: LCTYPE_CURRENCY LCTYPE_DATETIME LCTYPE_INT LCTYPE_FLOAT LCTYPE_NUMERIC LCTYPE_TEXT LCTYPE_BINARY
<i>count</i>	Long. Optional. Number of data values to be allocated for this field. The default is 1.

ClearVirtualCode method for LCField

This method clears the specified virtual code for the field.

Defined In

LCField

Syntax

Call *field*.ClearVirtualCode(*code*)

Parameters

code The code to clear. Zero (0) clears all codes.

Example

```
Option Public
Uselsx "*lsxlc"

Sub Initialize
    Dim session As New LCSession
    Dim field As New LCField (LCTYPE_INT)
    Dim Code As Long
    Dim text As String

    Call session.ListConnector (LCLIST_FIRST, , Code)
    Call field.SetVirtualCode (Code)
    While session.ListConnector (LCLIST_NEXT, , Code)
        Call field.SetVirtualCode (Code)
    Wend

    ' use the value to clear an individual connector's virtual code
    ' use 'zero' to clear all connectors' virtual codes

    Call field.ClearVirtualcode (0)
    Print "All of the virtual codes have been cleared."
End Sub
```

Example Output

All of the virtual codes have been cleared.

Compare method for LCField

Compares data values between two fields and returns the relationship.

Defined In

LCField

Syntax

Result = *thisField*.Compare(*thisIndex*, *baseField*, *baseIndex*, *valueCount*, *compareFlags*)

Parameters

<i>thisIndex</i>	Long. Index identifying the first value to be compared from <i>thisField</i> .
<i>baseField</i>	LCField. Base field for comparison.
<i>baseIndex</i>	Long. Index identifying the first base value in <i>baseField</i> .
<i>valueCount</i>	Long. Number of field values to be compared. Starting at the indicated index in each field, ValueCount consecutive fields from each fieldlist will be compared until all are compared or until there is a mismatch.
<i>compareFlags</i>	<p>Long. Zero or more of the following values, OR-ed together:</p> <p>LCCOMPAREF_NO_ZONE Do not consider timezone or daylight savings time information in datetime comparisons.</p> <p>LCCOMPAREF_SECOND Do not consider fractions of seconds in datetime comparisons.</p> <p>LCCOMPAREF_FLOAT_PREC Compare floating point values to only ten digits of precision.</p> <p>In addition, the following composite flag is supplied:</p>

Compare method for LCField

LCCOMPAREF_LOW_PREC

Composite of all low-precision LCCOMPAREF flags.

Return Value

Result

Result of the comparison:

Result > 0 (positive): The first mismatch that the function encountered in *thisField* is greater than the corresponding value in *baseField*.

Result < 0 (negative): The first mismatch that the function encountered in *thisField* is less than the corresponding value in *baseField*.

Result = 0: Each compared value in *thisField* is equal to the corresponding value in *baseField*.

Example

```
Option Public
Uselsx "**lsxc"
```

```
Sub Initialize
  Dim field1 As New LCField (LCTYPE_INT, 3)
  Dim field2 As New LCField (LCTYPE_INT, 2)

  Call field1.SetInt (1, 100)
  Call field1.SetInt (2, 300)
  Call field1.SetInt (3, 400)
  Call field2.SetInt (1, 300)
  Call field2.SetInt (2, 400)

  If (field1.Compare(2, field2, 1, 2, 0) <> 0) Then
    Print "the 2nd and 3rd values of field1 " & _
      "do not match the 1st and 2nd values of field2."
  Else
    Print "the 2nd and 3rd values of field1 " & _
      "match the 1st and 2nd values of field2."
  End If
End Sub
```

Example Output

the 2nd and 3rd values of field1 match the 1st and 2nd values of field2.

Convert method for LCField

This method converts data values to the data type of a destination field.

Defined In

LCField

Syntax

Call *thisField.Convert*(*thisIndex*, *srcField*, *srcIndex*, *valueCount*)

Parameters

<i>thisIndex</i>	Long. Index identifying the first target data value in <i>thisField</i> .
<i>srcField</i>	LCField. Source field for the data values. Values will be converted to the data type of <i>thisField</i> .
<i>srcIndex</i>	Long. Index identifying the first source data value to be converted in <i>SrcField</i> .
<i>valueCount</i>	Long. Number of field values to be converted. If the end of either field is encountered before converting this number of data values, then the function stops at that point.

Example

```
Option Public  
Uselsx "*lsxc"
```

```
Sub Initialize  
    Dim f_field As New LCField (LCTYPE_FLOAT)  
    Dim c_field As New LCField (LCTYPE_CURRENCY)  
  
    f_field.value = 12345.123456789  
    Call c_field.Convert (1, f_field, 1, 1)  
    Print "The float field is " & f_field.text(0) & " and it was "  
    Print "converted to a currency field as " & c_field.text(0)  
End Sub
```

Example Output

The float field is 12345.123456789 and it was
converted to a currency field as 12345.1235

Copy method for LCField

This method creates a field with the settings and values of a source field. The new field contains the same number of data values of the same data type as the source field. The new values are copies of, rather than references to, the original data.

Defined In

LCField

Syntax

Set *newField* = *origField*.Copy

Parameters

<i>origField</i>	LCField. The field object that you want to copy.
------------------	--

Return Value

<i>newField</i>	LCField. The copy of the original field.
-----------------	--

Example

```
Option Public  
Uselsx "**lsxc"
```

```
Sub Initialize  
  Dim field As New LCField (LCTYPE_FLOAT)  
  Dim c_field As LCField  
  
  field.value = 12345.123456789  
  Set c_field = field.Copy  
  Print "The float field is " & field.text(0)  
  Print "and its copy is " & c_field.text(0)  
End Sub
```

Example Output

```
The float field is 12345.123456789  
and its copy is 12345.123456789
```

GetCurrency method for LCField

This method retrieves a Currency data type from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

*Set newCurrency = lcField.***GetCurrency**(*index*)

Parameters

<i>index</i>	Long. Index identifying the field data value to be retrieved.
--------------	---

Return Value

<i>newCurrency</i>	The value of the retrieved data type.
--------------------	---------------------------------------

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim Fld As New LCField (LCTYPE_TEXT)  
    Fld.Text = "1234.56789"  
  
    Dim vCurr As LCCurrency  
    Set vCurr = Fld.GetCurrency (1)  
    Print "The Currency representation of the field is " & vCurr.Text  
End Sub
```

Example Output

The Currency representation of the field is 1234.5678

GetDatetime method for LCField

This method retrieves a Datetime data type from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

*Set newDatetime = lcField.***GetDatetime**(*index*)

Parameters

<i>index</i>	Long. Index identifying the field data value to be retrieved.
--------------	---

Return Value

<i>newDatetime</i>	The value of the retrieved data type.
--------------------	---------------------------------------

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim Fld As New LCField (LCTYPE_TEXT)  
    Fld.Text = "1234.56789"  
  
    Dim vDate As LCDateTime  
    Fld.Text = "10/12/1998"  
    Set vDate = Fld.GetDatetime (1)  
    Print "The Datetime representation of the field is " & vDate.Text  
End Sub
```

Example Output

The Datetime representation of the field is 10/12/1998

GetFieldlist method for LCField

This method retrieves a fieldlist from a field. The resulting fieldlist is a reference to the original inside the field. If the actual field data type is different, an `LCFAIL_INVALID_CONVERT` error will occur.

Defined In

LCField

Syntax

Set *newFieldList* = *lcField*.**GetFieldList**(*index*)

Parameters

<i>index</i>	Long. Index identifying the field data value to be retrieved.
--------------	---

Return Value

<i>newFieldList</i>	The value of the retrieved data type.
---------------------	---------------------------------------

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim Record As New LCFieldlist
    Dim SubRecord As New LCFieldlist
    Dim field As LCField
```

```
    REM start building FieldList
```

```
    Set field = Record.Append ("group", LCTYPE_INT)
```

```
    field.Value = 4200
```

```
    REM Build SubFieldList
```

```
    Set field = SubRecord.Append ("category", LCTYPE_TEXT)
```

```
    field.Value = "potato"
```

```
    Set field = SubRecord.Append ("description", LCTYPE_TEXT)
```

```
    field.Value = "russet"
```

```
    Set field = SubRecord.Append ("sku", LCTYPE_INT)
```

```
    field.Value = 4207
```

```
    REM return to building the FieldList
```

```
    Set field = Record.Append ("item", LCTYPE_FIELDLIST)
```

```
    REM now assign the SubRecord to the Record
```

```
    Call field.SetFieldList (1, SubRecord)
```

```
    Delete SubRecord
```

```
    Set SubRecord = field.getFieldList (1)
```

```
    Print "The first field of the field's fieldlist is " & SubRecord.Names(0)
```

```
End Sub
```

Example Output

The first field of the field's fieldlist is category

GetFloat method for LCField

This method retrieves a LotusScript double data type from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set *newFloat* = *lcField*.**GetFloat**(*index*)

Parameters

<i>index</i>	Long. Index identifying the field data value to be retrieved.
--------------	---

Return Value

<i>newFloat</i>	The value of the retrieved data type.
-----------------	---------------------------------------

Example

```
Option Public  
Uselsx "**lsxc"
```

```
Sub Initialize  
    Dim Fld As New LCField (LCTYPE_TEXT)  
    Fld.Text = "1234.56789"  
  
    Dim vFloat As Double  
    vFloat = Fld.GetFloat (1)  
    Print "The Float representation of the field is " & vFloat  
End Sub
```

Example Output

The Float representation of the field is 1234.56789

GetFormatDatetime method for LCField

This method retrieves the format of a Datetime type field. It is only valid for fields of type Datetime.

Use the LCField.SetFormatDatetime method to assign stream format information. Refer to the Field Format section for a description of format values.

Defined In

LCField

Syntax

Call *thisField*.GetFormatDatetime(*dateTimeFlags*, *size*)

Parameters

<i>dateTimeFlags</i>	Long. Optional. Datetime flags of the field. Refer to the Field Format Datetime section for a description of the flags.
<i>size</i>	Long. Optional. Size in bytes of the datetime field. Zero indicates the size of an a Lotus Connector Datetime (8 bytes).

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize
```

```
    Dim field As New LCField (LCTYPE_DATETIME, 1)
```

```
    Dim flags As Long
```

```
    Dim size As Long
```

```
    Call field.SetFormatDatetime (LCDATETIMEF_NO_TIME, 0)
```

```
    Call field.GetFormatDateTime (flags, size)
```

```
    Print "The datetime format flag setting is " & Hex(flags) & "h"
```

```
End Sub
```

Example Output

Error! Cannot open file.

GetFormatNumber method for LCField

This method retrieves the format of a number type field. It is only valid for fields of type int, currency, float or numeric.

Use the LCField.SetFormatNumber method to assign stream format information. Refer to the Field Format section for a description of format values.

Defined In

LCField

Syntax

Call *thisField*.GetFormatNumber(*numberFlags*, *size*, *precision*, *scale*)

Parameters

<i>numberFlags</i>	Long. Optional. Number flags of the number field. Refer to the Field Number Format section for a description of the flags. The default is Nothing.
<i>size</i>	Long. Optional. Size in bytes of the number field. Zero indicates the size of the corresponding number object (LONG, DOUBLE, LCCURRENCY, or LCNUMERIC). The default is Nothing.
<i>precision</i>	Long. Optional. Precision of the number field. Zero indicates not used for this field. The default is Nothing.
<i>scale</i>	Long. Optional. Scale of the number field. Zero indicates not used for this field. Since zero is also a valid scale value, a constant LCSCALE_ZERO is defined which can be used to indicate a zero scale. The default is Nothing.

Example

```
Option Public
Uselsx "**lsxlc"

Sub Initialize
    Dim field As New LCField (LCTYPE_NUMERIC)
    Dim flags As Long

    Call field.SetFormatNumber (LCNUMBERF_UNSIGNED, , 10, 4)
    Call field.GetFormatNumber (flags)
    Print "The number flag setting is " & Hex(flags) & "h"
End Sub
```

Example Output

The number flag setting is 1h

GetFormatStream method for LCField

This method retrieves the format of a stream type field. It is only valid for fields of type text or binary.

Use the LCField.SetFormatStream method to assign stream format information. Refer to the Field Format section for a description of format values.

Defined In

LCField

Syntax

Call *thisField*.GetFormatStream(*streamFlags*, *maxLength*, *streamFormat*)

Parameters

<i>streamFlags</i>	Long. Optional. Stream flags for the field. Refer to the Stream Class section for a description of the flags. The default is Nothing.
<i>maxLength</i>	Long. Optional. Maximum length of the stream field. A value of zero indicates no maximum length. The default is Nothing.
<i>streamFormat</i>	Long. Optional. Stream format of the stream field. A value of zero indicates no specified stream format. The default is Nothing.

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim field As New LCField (LCTYPE_BINARY)
```

```
    Dim fmt As Long
```

```
    Dim flags As Long
```

```
    Call field.SetFormatStream (LCSTREAMF_NO_CASE, 256, LCSTREAMFMT_BIG5)
```

```
    Call field.GetFormatStream (flags, , fmt)
```

```
    Print "The stream format and flag settings are:"
```

```
    Print "format=" & fmt & " flags=" & Hex(flags) & "h"
```

```
End Sub
```

Example Output

The stream format and flag settings are:

format=26 flags=10h

GetInt method for LCField

This method retrieves an integer from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

*Set newInt = thisField.***GetInt**(*index*)

Parameters

<i>index</i>	Long. Index identifying the field data value to be retrieved.
--------------	---

Return Value

<i>newInt</i>	The value of the retrieved data type.
---------------	---------------------------------------

Example

```
Option Public  
Usesxl "xlslc"
```

```
Sub Initialize  
    Dim fld As New LCField (LCTYPE_TEXT)  
    fld.Text = "1234.56789"  
  
    Dim vInt As Long  
    vInt = fld.GetInt (1)  
    Print "The Int representation of the field is " & vInt  
End Sub
```

Example Output

The Int representation of the field is 1234

GetNumeric method for LCField

This method retrieves a numeric value from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

*Set newNumeric = thisField.***GetNumeric**(*index*)

Parameters

<i>index</i>	Long. Index identifying the field data value to be retrieved.
--------------	---

Return Value

<i>newNumeric</i>	The value of the retrieved data type.
-------------------	---------------------------------------

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize  
    Dim Fld As New LCField (LCTYPE_TEXT)  
    Fld.Text = "1234.56789"  
  
    Dim vNumr As LCNumeric  
    Set vNumr = Fld.GetNumeric (1)  
    Print "The Numeric representation of the field is " & vNumr.Text  
End Sub
```

Example Output

The Numeric representation of the field is 1234.56789

GetStream method for LCField

This method retrieves a stream value from a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set *newStream* = *thisField*.**GetStream** (*index*, *streamFormat*)

Parameters

<i>index</i>	Long. Index identifying the field data value to be retrieved.
<i>streamFormat</i>	Long. Stream format to obtain the data in. Any valid stream format is allowed. In addition, a value of zero will either use the stream format of the field (if <i>thisField</i> is TEXT or BINARY), or convert to native text format LCSTREAMFMT_NATIVE (if <i>thisField</i> is not a local type).

Return Value

<i>newStream</i>	The value of the retrieved data type.
------------------	---------------------------------------

Example

```
Option Public  
Uselsx "*lsxc"
```

```
Sub Initialize  
    Dim Fld As New LCField (LCTYPE_TEXT)  
    Fld.Text = "1234.56789"  
  
    Dim vStrm As LCStream  
    Set vStrm = Fld.GetStream (1, LCSTREAMFMT_ASCII)  
    Print "The Stream representation of the field is " & vStrm.Text  
End Sub
```

Example Output

The Stream representation of the field is 1234.56789

LookupVirtualCode method for LCField

This method checks if a specific VirtualCode has been set for a field.

Defined In

LCField

Syntax

Flag = *field*.**LookupVirtualCode**(*virtualCode*)

Parameters

<i>virtualCode</i>	The virtual code to look for on the list of codes for this field.
--------------------	---

Return Value

<i>Flag</i>	TRUE if this virtual code is set for the field; FALSE otherwise.
-------------	--

Example

```
Option Public
Uselsx "*lsxc"
```

```
Sub Initialize
  Dim session As New LCSession
  Dim field As New LCField (LCTYPE_INT)
  Dim Code As Long
  Dim text As String

  Call session.ListConnector (LCLIST_FIRST, , Code)
  Call field.SetVirtualCode (Code)
  While session.ListConnector (LCLIST_NEXT, , Code)
    Call field.SetVirtualCode (Code)
  Wend

  If (field.LookupVirtualCode (&H10000)) Then
    Print "The field has virtual code 10000h set."
  Else
    Print "The field does not have virtual code 10000h set."
  End If
End Sub
```

Example Output

The field has virtual code 10000h set.

SetCurrency method for LCField

This method assigns the currency value to the specified data index of the field. If the field data type is different, conversion will be attempted.

Defined In

LCField

Syntax

Call *thisField.SetCurrency(index, srcCurrency)*

Parameters

<i>index</i>	Long. Index identifying the value that is to be assigned.
<i>srcCurrency</i>	LCCurrency. Value to be assigned to the field data value.

Example

```
Option Public  
Uselsx "**lsxc"
```

```
Sub Initialize  
    Dim field As New LCField (LCTYPE_TEXT)  
    Dim number As New LCCurrency  
  
    number.value = 1234567890.1234  
    Call field.SetCurrency (1, number)  
    Print "The field's value is " & field.text(0)  
End Sub
```

Example Output

The field's value is 1234567890.1234

SetDatetime method for LCField

This method assigns the datetime value to the specified index of the field. If the field data type is different, conversion will be attempted.

Defined In

LCField

Syntax

Call *thisField.SetDateTime(index, srcDateTime)*

Parameters

<i>index</i>	Long. Index identifying the value that is to be assigned.
<i>srcDateTime</i>	LCDatetime. Value to be assigned to the field data value.

Example

```
Option Public  
Uselsx "**lsxc"
```

```
Sub Initialize  
    Dim field As New LCField (LCTYPE_TEXT)  
    Dim clock As New LCDatetime  
  
    clock.SetCurrent  
    Call field.SetDatetime (1, clock)  
    Print "The field's value is " & field.text(0)  
End Sub
```

Example Output

The field's value is 09/08/1998 05:22:30.86 PM

SetFieldlist method for LCField

This method assigns the fieldlist value to the specified index of the field. If the field data type is different, an LCFAIL_INVALID CONVERT error will occur.

Defined In

LCField

Syntax

Call *thisField.SetFieldlist(index, srcFieldlist)*

Parameters

<i>index</i>	Long. Index identifying the value that is to be assigned.
<i>srcFieldlist</i>	LCFieldlist. Value to be assigned to the field data value.

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
  Dim Record As New LCFieldList
  Dim SubRecord As New LCFieldList
  Dim field As LCField

  REM start building FieldList
  Set field = Record.Append ("group", LCTYPE_INT)
  field.Value = 4200
  REM Build SubFieldList
  Set field = SubRecord.Append ("category", LCTYPE_TEXT)
  field.Value = "potato"
  Set field = SubRecord.Append ("description", LCTYPE_TEXT)
  field.Value = "russet"
  Set field = SubRecord.Append ("sku", LCTYPE_INT)
  field.Value = 4207
  REM return to building the FieldList
  Set field = Record.Append ("item", LCTYPE_FIELDLIST)
  REM now assign the SubRecord to the Record
  Call field.SetFieldList (1, SubRecord)
  REM Take a look at the Record while debugging the LotusScript
  Print "The sub fieldlist has successfully been appended to parent fieldlist as another field."
End Sub
```

Example Output

The sub fieldlist has successfully been appended to parent fieldlist as another field.

SetFloat method for LCField

This method assigns a value to a field of type float. If the field data type is different, conversion will be attempted.

Defined In

LCField

Syntax

Call *thisField.SetFloat(index, srcFloat)*

Parameters

<i>index</i>	Long. Index identifying the value that is to be assigned.
<i>srcFloat</i>	Double. Value to be assigned to the field data value.

Example

```
Option Public  
Uselsx "**lsxc"
```

```
Sub Initialize  
    Dim field As New LCField (LCTYPE_TEXT)  
    Dim number As Double  
  
    number = Pi  
    Call field.SetFloat (1, number)  
    Print "The field's value is " & field.text(0)  
End Sub
```

Example Output

The field's value is 3.14159265358979

SetFormatDatetime method for LCField

This method assigns the current format setting for a datetime field. Calling this method clears all values for this field.

Defined In

LCField

Syntax

Call *thisField.SetFormatDatetime(datetimeFlags, size)*

Parameters

datetimeFlags

Long. Optional. Datetime flags assigned to the field. Refer to the Field Datetime Format section for a description of the flags. The default is 0.

The *datetimeFlags* do not affect the data. As with the *size* value, it is only relevant when interacting with the external system. The flags will have no effect on the conversion of a field's value to or from a stream or text.

size

Long. Optional. Size in bytes assigned to the datetime field. Zero indicates the size of an Lotus Connector Datetime (8 bytes). The default is 0.

The *size* value is used for metadata creation by data systems which support datetime datatypes of different sizes. For systems which support only a single datetime datatype, the value of *size* is ignored. The default value for *size* is zero.

For example, Sybase supports both a 4 byte datetime as well as an 8 byte version. When building up a fieldlist that will be used to create a Sybase table (see `LCConnection.Create` for an example), use the `SetFormatDatetime` method to indicate which size datetime datatype should be used.

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim field As New LCField (LCTYPE_DATETIME, 1)  
    Dim flags As Long  
    Dim size As Long  
  
    Call field.SetFormatDatetime (LCDATETIMEF_NO_TIME, 0)  
    Call field.GetFormatDateTime (flags, size)  
    Print "The datetime format flag setting is " & Hex(flags) & "h"  
End Sub
```

Example Output

The datetime format flag setting is 2h

SetFormatNumber method for LCField

This method assigns the current format setting for a number field. Calling this method clears all values for this field.

Defined In

LCField

Syntax

Call *thisField.SetFormatNumber* (*numberFlags*, *size*, *precision*, *scale*)

Parameters

<i>numberFlags</i>	Long. Optional. Number flags assigned to the number field. Refer to the Field Number Format section for a description of the flags. The default is 0.
<i>size</i>	Long. Optional. Size in bytes assigned to the number field. Zero indicates the size of the corresponding number object (LONG, DOUBLE, LCCURRENCY, or LCNUMERIC). The default is 0.
<i>precision</i>	Long. Optional. Precision assigned to the number field. Zero indicates not used for this field. The default is LCMAX_NUMERIC_PREC.
<i>scale</i>	Long. Optional. Scale assigned to the number field. Zero indicates not used for this field. Since zero is also a valid scale value, a constant LCSCALE_ZERO is defined which can be used to indicate a zero scale. The default is LCMAX_NUMERIC_PREC / 2.

Example

```
Option Public
Uselsx "**lsxlc"

Sub Initialize
    Dim field As New LCField (LCTYPE_NUMERIC)
    Dim flags As Long

    Call field.SetFormatNumber (LCNUMBERF_UNSIGNED, , 10, 4)
    Call field.GetFormatNumber (flags)
    Print "The number flag setting is " & Hex(flags) & "h"
End Sub
```

Example Output

The number flag setting is 1h

SetFormatStream method for LCField

This method assigns the current format setting for a stream field. Calling this method clears all values for this field.

Defined In

LCField

Syntax

Call *thisField*.**SetFormatStream**(*streamFlags*, *maxLength*, *streamFormat*)

Parameters

<i>streamFlags</i>	Long. Optional Stream flags assigned to the field. Refer to the Stream Class section for a description of the flags. The default is 0.
<i>maxLength</i>	Long. Optional Maximum length assigned to the stream field. A value of zero indicates no maximum length. The default is 0.
<i>streamFormat</i>	Long. Optional Stream format assigned to the stream field. A value of zero indicates no specified stream format. The default is LCSTREAMFMT_UNICODE.

Example

```
Option Public
Uselsx "**lsxc"
```

```
Sub Initialize
```

```
    Dim field As New LCField (LCTYPE_BINARY)
```

```
    Dim fmt As Long
```

```
    Dim flags As Long
```

```
    Call field.SetFormatStream (LCSTREAMF_NO_CASE, 256, LCSTREAMFMT_BIG5)
```

```
    Call field.GetFormatStream (flags, , fmt)
```

```
    Print "The stream format and flag settings are: format=" & fmt & " flags=" & Hex(flags) & "h"
End Sub
```

Example Output

The stream format and flag settings are: format=26 flags=10h

SetInt method for LCField

This method assigns a value to a field of type long. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Call *thisField.SetInt* (*index*, *srcInt*)

Parameters

<i>index</i>	Long. Index identifying the value that is to be assigned.
<i>srcInt</i>	Long. Value to be assigned to the field data value.

Example

```
Option Public  
Uselsx "*lsxc"
```

```
Sub Initialize  
    Dim field As New LCField (LCTYPE_TEXT)  
    Dim number As Long  
  
    number = Pi  
    Call field.SetInt (1, number)  
    Print "The field's value is " & field.text(0)  
End Sub
```

Example Output

The field's value is 3

SetNumeric method for LCField

This method assigns a value to a field of type LCNumeric. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Call *thisField.SetNumeric(index, srcNumeric)*

Parameters

<i>index</i>	Long. Index identifying the value that is to be assigned.
<i>srcNumeric</i>	LCNumeric. Value to be assigned to the field data value.

Example

```
Option Public  
Uselsx "**lsxc"
```

```
Sub Initialize  
    Dim field As New LCField (LCTYPE_TEXT)  
    Dim number As New LCNumeric  
  
    number.value = 1234567890.12345678  
    Call field.SetNumeric (1, number)  
    Print "The field's value is " & field.text(0)  
End Sub
```

Example Output

The field's value is 1234567890.12346

SetStream method for LCField

This method assigns a value to a field of type LCStream (text or binary). If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Call *thisfield*.**SetStream**(*index*, *srcStream*)

Parameters

<i>index</i>	Long. Index identifying the value that is to be assigned.
<i>srcStream</i>	LCStream. Value to be assigned to the field data value.

Example

```
Option Public
Uselsx "**lsxc"

Sub Initialize
    Dim field As New LCField (LCTYPE_TEXT)
    Dim msg As New LCStream

    msg.Text = "Hello World"
    Call field.SetStream (1, msg)
    Print "The field's value is " & field.text(0)
End Sub
```

Example Output

The field's value is Hello World

SetVirtualCode method for LCField

This method adds a virtual code to the list of virtual codes for a field.

Defined In

LCField

Syntax

Call *thisField.SetVirtualCode(virtualCode)*

Parameters

<i>virtualCode</i>	Long. VirtualCode value to assign to <i>thisField</i> .
--------------------	---

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
  Dim session As New LCSession
  Dim field As New LCField (LCTYPE_INT)
  Dim Code As Long
  Dim text As String

  Call session.ListConnector (LCLIST_FIRST, , Code)
  Call field.SetVirtualCode (Code)
  text = Hex(Code) & "h"
  While session.ListConnector (LCLIST_NEXT, , Code)
    Call field.SetVirtualCode (Code)
    text = text & ", " & Hex(Code) & "h"
  Wend
  Print "The field virtual codes (one per connector) are " & text
End Sub
```

Example Output

The field virtual codes (one per connector) are 10000h, 20000h, 30000h, 40000h, 50000h

Chapter 6

LCFieldlist Class

This chapter provides information about the LCFieldlist class methods and properties.

Overview

The LCFieldlist class represents metadata (the description of data from a data source) for a record and may reference data, in the form of LCFields, for one or more record values. A fieldlist is a list of fields and field names with a represented order. Fields within a fieldlist can be added, modified, retrieved, or listed in multiple ways. The fields, field names, and order are separate entities and only have relation within an individual fieldlist.

Note that many situations exist where the names or order must be changed to accommodate different connections, but the data needs to remain constant. In these cases, more than one fieldlist may be created, each referenced by the same fields using the Map, MapName, or Merge methods.

Accessing Field Data

Field data within an LCFieldlist object may be accessed as a property by name. The datatype of the property depends on the datatype of the field. The property is an array of LotusScript datatypes, as shown in the table below. For example, if a fieldlist has a field named OrderID, its first value may be obtained with the script:

```
Identification = LCFieldlist.OrderID(0).
```

Likewise the field may be set with the syntax:

```
LCFieldlist.OrderID = "Bart-001".
```

The LCFieldlist object may contain one or more values for each field contained within, depending on the parameters to the constructor. If the object is constructed for more than a single value, subsequent values may be accessed through the field names by changing the subscript. Multiple values may be set by assigning an array.

Field Type	Value Return Type
LCTYPE_CURRENCY	Currency
LCTYPE_DATETIME	Variant (Date/Time)
LCTYPE_INT	Long
LCTYPE_FLOAT	Double
LCTYPE_NUMERIC	Double
LCTYPE_TEXT	String
LCTYPE_BINARY	String

Fieldlist Merging

Fieldlist merging with `LCFieldlistMerge` and `LCFieldlistMergeVirtual` support field mapping. To perform mapping, two fieldlists are provided, one as the name source and the other as the field/data source. Depending on merge options, a mapping is produced between the names of the name source and the fields of the data source, and a third fieldlist is produced which references parts of the original fieldlists. When using virtual fields and `LCFieldlistMergeVirtual`, fields in the data fieldlist which match the supplied virtual code are excluded from mapping and are added to a separate new virtual fieldlist.

Mapping and Merging

When a fieldlist is to be used for more than one specific action, there is often a need to have it treated differently by each of those actions. For example, fields in a fieldlist map need to be reordered, renamed, or removed, but only in the context of one particular operation. For such situations, four methods are provided to perform this mapping: `Map`, `MapName`, `Merge`, and `MergeVirtual`. `MergeVirtual` is simply a variation of `Merge` which handles virtual codes as well. These methods produce a new fieldlist from an existing fieldlist, but with the same fields referenced from both fieldlists, allowing two different 'maps' into a set of fields. If these two fieldlists are sent to different Connector methods, then the Connectors will see different sets of fields in a different order, but referring to the same data and common fields.

The three methods `Map`, `MapName`, and `Merge` have various levels of difficulty and control available to the caller; which one is needed depends on the specific situation. In general, the simplest method which supports your requirements should be used. These methods are listed below, starting with the simplest:

`Map` is applicable when the names in the fieldlist are correct, but fields either need to be reordered or removed. It accepts a fieldlist and a text list of names, and the new fieldlist contains the fields from the original fieldlist reordered in the order they are listed in the text list. In addition, any fields not included in the text list are excluded from the new fieldlist.

MapName adds renaming of fields to Map, but is otherwise identical. By adding a second text list of names, it will rename each field in the first text list with the corresponding entry in the second text list.

Merge accepts two fieldlists - one which provides the list of fields, and one which provides the list of names. In addition, it accepts a set of flags which further control the new fieldlist produced. These flags control the type of mapping, whether to allow fields to be excluded from either fieldlist, and how to interact with specific field flags. Merge should only be used when the functionality of Map and MapName is insufficient for your needs, as it is much more complex to use.

LCFieldlist Class Methods Summary

Creation

The following methods create and free Fieldlist instances. The Fieldlist Merging methods also create new fieldlists.

New LCFieldlist	(Constructor) Creates a new Fieldlist object instance.
LCFieldlist.Copy	Creates a new Fieldlist object instance as a copy of another fieldlist. The field data is also copied.
LCFieldlist.CopyRef	Creates a new Fieldlist object instance as a partial copy of another fieldlist. The contained fields are not copied, but are rather referenced.
Delete LCFieldlist	(Destructor) Frees a Fieldlist object instance created with the constructor, LCFieldlist.Copy, LCFieldlist.CopyRef, LCFieldlist.Merge, or LCFieldlist.MergeVirtual.

Fieldlist Retrieval

The following methods support retrieval of fieldlist contents.

LCFieldlist.GetField	Retrieves a particular field from a fieldlist by field index.
LCFieldlist.GetName	Retrieves a copy of a field name from a fieldlist by field index.
LCFieldlist.Lookup	Locates a field in a fieldlist by field name.

Fieldlist Modification

The following methods support modification of fieldlist contents.

LCFieldlist.Append	Adds a new field to the end of a fieldlist.
LCFieldlist.Insert	Inserts a new field into a fieldlist.
LCFieldlist.Replace	Replaces a field in a fieldlist with a new field.

<code>LCFieldlist.Remove</code>	Removes a field from a fieldlist
<code>LCFieldlist.SetName</code>	Renames a field within a fieldlist.

Fieldlist Merging

The following methods support fieldlist merging.

<code>LCFieldlist.Merge</code>	Merges a name fieldlist and data fieldlist to produce a mapped fieldlist.
<code>LCFieldlist.MergeVirtual</code>	Merges a name fieldlist and data fieldlist to produce a mapped fieldlist. Fields with matching virtual codes are added to a new virtual fieldlist.

Fieldlist Mapping

<code>LCFieldlist.Map</code>	Maps fields with the same names but changes the positions and can exclude some fields.
<code>LCFieldlist.MapName</code>	Maps fields with different names.

LCFieldlist Properties

FieldCount	Number of fields in the fieldlist. [Read-Only]
Fields As Variant	An array of LCFields. [Read-Only]
Names As Variant	An array of strings.
RecordCount As Long	Number of records per field. [Read-Only]
Sequence As Long	A fieldlist sequence number. [Read-Only]
All other properties are dynamic. Field data may be accessed by referencing the field name.	

New method for LCFieldlist

This is the constructor for LCFieldlist.

Defined In

LCFieldlist

Syntax

Dim *variableName* **as New LCFieldlist**(*recordCount*, *fieldFlags*)

Parameters

recordCount

Long. Optional. The number of records in the fieldlist. The default is 1. All fields added to the fieldlist will have this number of records.

fieldFlags

Long. Optional. The default field flags for the fieldlist (zero or more LCFIELD_XXX flags ORed together). These are set as the initial field flags for each field added to the fieldlist. The default is 1.

Append method for LCFieldlist

This method appends a field to an existing fieldlist.

Defined In

LCFieldlist

Syntax

Set *field* = *fieldlist*.Append(*fieldName*, *dataType*)

Parameters

<i>fieldName</i>	String. The name for the field.
<i>dataType</i>	Long. The constant for the datatype. One of the following: LCTYPE_INT LCTYPE_FLOAT LCTYPE_CURRENCY LCTYPE_NUMERIC LCTYPE_DATETIME LCTYPE_TEXT LCTYPE_BINARY LCTYPE_FIELDLIST LCTYPE_CONNECTION

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    REM this example copies a DB2 table
```

```
    Dim src As New LCConnection ("db2")
```

```
    Dim fldLstRecord As New LCFieldList
```

```
    Dim fld As LCField
```

```
    REM build the table definition
```

```
    Call fldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
```

```
    Call fldLstRecord.Append ("CONTACTNAME", LCTYPE_TEXT)
```

```
    Call fldLstRecord.Append ("COMPANYNAME", LCTYPE_TEXT)
```

```
    REM set properties to connect to both data sources
```

```
    src.Database = "Gold"
```

```
    src.Userid = "JDoe"
```

```
    src.Password = "xyzyzy"
```

```
    src.Metadata = "customer"
```

```
    REM now connect
```

```
    src.Connect
```

```
    REM create it based on the metadata property already set above
```

```
    On Error LCFAIL_DUPLICATE Goto tableexists
```

```
    Call src.Create (LCOBJECT_METADATA, fldLstRecord)
```

```
    Print "The '" & src.Metadata & "' table did not exist so it was created."
```

```
    End
```

```
tableexists:
```

```
    Print "The '" & src.Metadata & "' table exists."
```

```
    End
```

```
End Sub
```

Example Output

The 'customer' table exists.

Copy method for LCFieldlist

This method creates a duplicate copy of an LCFieldlist and all its data.

Defined In

LCFieldlist

Syntax

Set fldListRecordNew = fldListRecord.Copy

Parameters

<i>fldListRecord</i>	LCFieldlist. The fieldlist that you want to copy.
----------------------	---

Return Value

<i>fldListRecordNew</i>	LCFieldlist. The copy of the <i>fldListRecord</i> object.
-------------------------	---

Example

```

Option Public
Uselsx "*lsxc"

Sub Initialize
    Dim olist As New LCFieldlist
    Dim nlist As LCFieldlist
    Dim ofield As LCField

    Set ofield = olist.Append ("FirsName", LCTYPE_TEXT)
    ofield.Text = "Chi Len"

    Set nlist = olist.Copy
    Set nfield = nlist.GetField (1)

    Call olist.SetName (1, "FullName")
    ofield.Text = "Cheiko"

    Print "The copy contains:"
    Print "field named " & nlist.Names(0) & " whose value is " & nfield.Text(0)
End Sub

```

Example Output

The copy contains:
field named FirsName whose value is Chi Len

CopyField method for LCFieldlist

This method copies an existing field within a fieldlist at a specified position.

Defined In

LCFieldlist

Syntax

Set newField = fieldList.CopyField(index, srcField, name)

Parameters

<i>index</i>	Long. Position of the field to copy.
<i>srcField</i>	LCField. The source field to copy.
<i>name</i>	String. The name to give the copy of the field.

Return Value

<i>name</i>	String. The name of the copy of the field.
-------------	--

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
    Dim fldLstRecord As New LCFieldList
    Dim fld As New LCField (LCTYPE_TEXT)
    Dim ref As LCField
    Dim text As String

    ' There are a number of ways to build a fieldlist
    ' Append will add a field to a list given a type
    Set ref = FldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
    Set ref = FldLstRecord.Append ("COMPANYID", LCTYPE_INT)

    ' Insert is like Append but the position within
    ' the fieldlist must be specified
    Set ref = FldLstRecord.Insert (1, "COMPANYADDRESS", LCTYPE_TEXT)

    ' CopyField will add a field to a list
    ' using an existing field as a template
    fld.Flags = LCFIELD_KEY
    Set ref = FldLstRecord.CopyField (1, fld, "CONTACTNAME")

    ' IncludeField will add an existing field to a list,
    ' making it part of the list. in this case, 'fld'
    ' becomes a reference into the fieldlist
    fld.Flags = 0
    Call FldLstRecord.IncludeField (3, fld, "COMPANYCITY")

    text = ""
    Forall fieldname In FldLstRecord.Names
        If Text = "" Then text = fieldname Else text = text + ", " + fieldname
    End Forall
    Print "The field list looks like: " & text
End Sub
```

Example Output

The field list looks like: CONTACTNAME, COMPANYADDRESS, COMPANYCITY, ACCOUNTMANAGER, COMPANYID

CopyRef method for LCFieldlist

Creates a new fieldlist object instance as a partial copy of another fieldlist. The fields in the original fieldlist are not copied, but are referenced.

This method creates a new fieldlist with separate name space but with the same fields and data space as an existing fieldlist. The field names are copied, but the new fieldlist references the original fields. In this situation, any changes to the data of one fieldlist is actually a change to the data for all fieldlists referencing the same data space.

To copy a fieldlist's metadata and data, use LCFieldlistCopy.

Defined In

LCFieldlist

Syntax

Set *fldListRecordNew* = *fldListRecord*.**CopyRef**

Parameters

<i>fldListRecord</i>	LCFieldlist. The fieldlist that you want to make a reference copy of.
----------------------	---

Return Values

<i>fldListRecordNew</i>	LCFieldlist. The reference copy of the fieldlist.
-------------------------	---

Example

```

Option Public
Uselsx "*lsxlc"

Sub Initialize
    Dim olist As New LCFieldlist
    Dim nlist As LCFieldlist
    Dim ofield As LCField

    Set ofield = olist.Append ("FirsName", LCTYPE_TEXT)
    ofield.Text = "Chi Len"

    Set nlist = olist.CopyRef
    Set nfield = nlist.GetField (1)

    Call olist.SetName (1, "FullName")
    ofield.Text = "Cheiko"

    Print "The copy contains:"
    Print "field named " & nlist.Names(0) & " whose value is " & nfield.Text(0)
End Sub

```

Example Output

The copy contains:
field named FirsName whose value is Cheiko

GetField method for LCFieldlist

This method gets a field reference from a fieldlist result set and places its value into a variable.

Defined In

LCFieldlist

Syntax

*Set field = fldLstRecord.***GetField**(*index*)

Parameters

<i>index</i>	Long. Position of the field to be returned.
--------------	---

Return Values

<i>field</i>	LCField. The field at the index position in the fieldlist.
--------------	--

Example

```
Option Public
Uselsx "*lsxc"
```

```
Sub Initialize
```

```
    Dim connect As New LCConnection ("db2")
```

```
    Dim conFldLst As New LCFieldList
```

```
    Dim field As LCField
```

```
    ' this section assigns the appropriate properties to connect to DB2
```

```
    connect.Database = "Gold"
```

```
    connect.Userid = "JDoe"
```

```
    connect.Password = "xyzzzy"
```

```
    connect.Metadata = "customer"
```

```
    ' connect to DB2
```

```
    connect.Connect
```

```
    ' now perform the catalog action - in this case for fields
```

```
    If (connect.Catalog (LCOBJECT_FIELD, conFldLst) = 0) Then
```

```
        Print "No tables were found."
```

```
    Else
```

```
        Set field = conFldLst.GetField(1)
```

```
        Print "The columns in the " & connect.Metadata & " table include:"
```

```
        While (connect.Fetch (conFldLst) > 0)
```

```
            Print "    " & field.text(0)
```

```
        Wend
```

```
    End If
```

```
End Sub
```

Example Output

The columns in the 'CUSTOMER' table include:

ACCOUNTMANAGER

CONTACTNAME

COMPANYNAME

COMPANYADDRESS

COMPANYCITY

COMPANYSTATE

COMPANYPHONE

GetName method for LCFieldlist

This method obtains a copy of the name of a field within a fieldlist.

Defined In

LCFieldlist

Syntax

fieldName = *fieldListRecord*.**GetName**(*index*)

Parameters

<i>fieldListRecord</i>	String. The fieldlist from which to retrieve the fieldname.
<i>index</i>	Long. The position of the field in the fieldlist.

Return Values

<i>fieldName</i>	String. The name of the field.
------------------	--------------------------------

Example

```

Option Public
Uselsx "**lsxc"

Sub Initialize
  Dim connect As New LCCConnection ("db2")
  Dim FldLst As New LCFieldlist
  Dim field As LCField
  Dim i As Long

  ' this section assigns the appropriate properties to connect to DB2
  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "xyzyzy"
  connect.Metadata = "customer"

  connect.Connect

  ' now perform the catalog action - in this case for fields
  If (connect.Select (Nothing, 1, FldLst) = 0) Then
    Print "The customer table was not found."
  Else
    Print "There are " & FldLst.FieldCount & _
      " columns in the " & connect.Metadata & " table."
    Print "They are:"
    For i = 1 To FldLst.FieldCount
      Print Tab(4); FldLst.GetName (i)
    Next
  End If
End Sub

```

Example Output

There are 7 columns in the 'customer' table.
They are:

ACCOUNTMANAGER
CONTACTNAME
COMPANYNAME
COMPANYADDRESS
COMPANYCITY
COMPANYSTATE
COMPANYPHONE

IncludeField method for LCFieldlist

This method includes an existing individual field into a fieldlist.

Defined In

LCFieldlist

Syntax

Call *fldLstRecord.IncludeField(index, field, fieldName)*

Parameters

<i>index</i>	Long. Position of the field within <i>fldLstRecord</i> .
<i>field</i>	LCField. The new field.
<i>fieldName</i>	String. The name of the new field.

Example

```
Option Public
Option Explicit
Uselsx "xlslc"
```

```
Sub Initialize
    Dim fldLstRecord As New LCFieldList
    Dim fld As New LCField (LCTYPE_TEXT)
    Dim ref As LCField
    Dim text As String

    ' There are a number of ways to build a fieldlist
    ' Append will add a field to a list given a type
    Set ref = FldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
    Set ref = FldLstRecord.Append ("COMPANYID", LCTYPE_INT)

    ' Insert is like Append but the position
    ' within the fieldlist must be specified
    Set ref = FldLstRecord.Insert (1, "COMPANYADDRESS", LCTYPE_TEXT)

    ' CopyField will add a field to a list
    ' using an existing field as a template
    fld.Flags = LCFIELD_KEY
    Set ref = FldLstRecord.CopyField (1, fld, "CONTACTNAME")

    ' IncludeField will add an existing field to a list,
    ' making it part of the list. in this case, 'fld'
    ' becomes a reference into the fieldlist
    fld.Flags = 0
    Call FldLstRecord.IncludeField (3, fld, "COMPANYCITY")

    text = ""
    Forall fieldname In FldLstRecord.Names
        If Text = "" Then text = fieldname Else text = text + ", " + fieldname
    End Forall
    Print "The field list looks like: " & text
End Sub
```

Example Output

The field list looks like: CONTACTNAME, COMPANYADDRESS, COMPANYCITY, ACCOUNTMANAGER, COMPANYID

Insert method for LCFieldlist

This method inserts a new field into an existing fieldlist at a specified index position.

Defined In

LCFieldlist

Syntax

Set fieldNew = fldLstRecord.Insert (index, fieldName, dataType)

Parameters

<i>index</i>	Long. Index number of the field before which the new field is inserted. An index number equal to the number of fields currently in the fieldlist (plus one) is equivalent to using LCFieldlist.Append.
<i>fieldName</i>	String. Name of the new field.
<i>dataType</i>	Long. Data type of the new field. One of the following: LCTYPE_INT LCTYPE_FLOAT LCTYPE_CURRENCY LCTYPE_NUMERIC LCTYPE_DATETIME LCTYPE_TEXT LCTYPE_BINARY LCTYPE_FIELDLIST LCTYPE_CONNECTION

Example

```
Option Public
Option Explicit
Uselsx "xlxc"
```

```
Sub Initialize
```

```
    Dim fldLstRecord As New LCFieldList
    Dim fld As New LCField (LCTYPE_TEXT)
    Dim ref As LCField
    Dim text As String
```

```
    ' There are a number of ways to build a fieldlist
    ' Append will add a field to a list given a type
    Set ref = fldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
    Set ref = fldLstRecord.Append ("COMPANYID", LCTYPE_INT)

    ' Insert is like Append but the position
    ' within the fieldlist must be specified
    Set ref = fldLstRecord.Insert (1, "COMPANYADDRESS", LCTYPE_TEXT)

    ' CopyField will add a field to a list
    ' using an existing field as a template
    fld.Flags = LCFIELD_KEY
    Set ref = fldLstRecord.CopyField (1, fld, "CONTACTNAME")

    ' IncludeField will add an existing field to a list,
    ' making it part of the list. in this case, 'fld'
    ' becomes a reference into the fieldlist
    fld.Flags = 0
    Call fldLstRecord.IncludeField (3, fld, "COMPANYCITY")
```

```
    text = ""
    Forall fieldname In fldLstRecord.Names
        If Text = "" Then text = fieldname Else text = text + ", " + fieldname
    End Forall
    Print "The field list is: " & text
End Sub
```

Example Output

The field list is: CONTACTNAME, COMPANYADDRESS, COMPANYCITY,
ACCOUNTMANAGER, COMPANYID

List method for LCFieldlist

This method iterates through fields in a fieldlist, optionally returning information.

Defined In

LCFieldlist

Syntax

Call *fldLstRecod*.**List**(*position*, *destField*, *index*, *dataType*, *flags*, *fieldName*)

Parameters

<i>position</i>	Long. Constant indicating whether to return the first or next Connector property. LCLIST_FIRST Return the first property in the property list. LCLIST_NEXT Return the next property (or the first property if this is the first call to this function for this Connection).
<i>destField</i>	LCField. Optional. The field in the fieldlist.
<i>index</i>	Long. Optional. Index of <i>destField</i> in the fieldlist. This may not advance consecutively for fieldlists produced with LCFieldlist.Merge or LCFieldlist.MergeVirtual.
<i>dataType</i>	Long. Optional. Data type of <i>destField</i> .
<i>flags</i>	Long. Optional. Field flags of <i>destField</i> . Refer to the Field Class section for a description of the flags.
<i>fieldName</i>	String. Optional. Copy of the name of <i>destField</i> .

Example

```
Option Public
Uselsx "*Isxlc"
```

```
Sub Initialize
```

```
Dim connect As New LCConnection ("db2")
Dim FldLst As New LCFieldlist
Dim pos As Long
Dim dtype As Long
Dim flags As Long
Dim fieldname As String
```

```
REM this section assigns the appropriate properties to connect to DB2
```

```
connect.Database = "Gold"
connect.Userid = "JDoe"
connect.Password = "xyzyzy"
connect.Metadata = "customer"
```

```
REM connect to DB2
```

```
connect.Connect
```

```
REM now perform the catalog action - in this case for fields
```

```
If (connect.Select (Nothing, 1, FldLst) = 0) Then
```

```
Print "The customer table was not found."
```

```
Else
```

```
Print "The table description is:"
```

```
pos = LCLIST_FIRST
```

```
While (FldLst.List (pos, , dtype, flags, fieldname) = True)
```

```
Print "  " + fieldname + " is type #" + _  
Cstr(dtype) + " with flags " + Hex(flags)
```

```
pos = LCLIST_NEXT
```

```
Wend
```

```
End If
```

```
End Sub
```

Example Output

The table description is:

ACCOUNTMANAGER is type #1 with flags 2
CONTACTNAME is type #6 with flags 2
COMPANYNAME is type #6 with flags 2
COMPANYADDRESS is type #6 with flags 2
COMPANYCITY is type #6 with flags 2
COMPANYSTATE is type #6 with flags 2
COMPANYPHONE is type #6 with flags 2

Lookup method for LCFieldlist

This method locates a field from a fieldlist based on field name.

Defined In

LCFieldlist

Syntax

Set field = fldListRecord.Lookup (fieldName, index)

Parameters

<i>fieldName</i>	String. Name of the field to look up.
------------------	---------------------------------------

Return Value

<i>index</i>	Long. Optional. Position of the field.
--------------	--

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
  Dim connect As New LCCConnection ("db2")
  Dim fldLst As New LCFieldlist
  Dim field As LCField
  Dim pos As Long
```

```
  REM this section assigns the appropriate properties to connect to DB2
```

```
  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "xyzyzy"
  connect.Metadata = "customer"
```

```
  REM connect to DB2
```

```
  connect.Connect
```

```
  REM now perform the catalog action - in this case for fields
```

```
  If (connect.Select (Nothing, 1, fldLst) = 0) Then
```

```
    Print "The customer table was not found."
```

```
  Else
```

```
    Set field = fldLst.Lookup ("contactname", pos)
```

```
    If Not (field Is Nothing) Then
```

```
      Print "Found 'ContactName' in the fieldlist"
```

```
      Print "at position " & pos
```

```
    Else
```

```
      Print "Did not find 'ContactName' in the fieldlist."
```

```
    End If
```

```
  End If
```

```
End Sub
```

Example Output

```
Found 'ContactName' in the fieldlist
at position 2
```

Map method for LCFieldlist

This method remaps fields in a fieldlist.

Use this method for field mapping operations when the fieldnames are the same in source and target, but you need to change the order of the fields or you wish to exclude some fields.

Defined In

LCFieldlist

Syntax

Call *fldListRecord*.**Map**(*baseFieldList*, *nameList*)

Parameters

baseFieldList LCFieldlist. The fieldlist to map.

nameList String. Comma-delimited names of fields from *baseFieldlist* in the new, remapped order.

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
  Dim session As New LCSession
  Dim srcCon As New LCConnection ("db2")
  Dim fldLst As New LCFieldList
  Dim fetchLst As New LCFieldList
  Dim count As Long
  Dim cname As LCField
  Dim ccity As LCField
  Dim cstate As LCField
```

```
  ' set the appropriate properties to connect to the data sources
  srcCon.Database = "Gold"
  srcCon.Userid = "JDoe"
  srcCon.Password = "xyzyzy"
  srcCon.Metadata = "customer"
```

```
  srcCon.Connect
```

```
  ' perform a select and get the records with
  ' the fields wanted, in our fldLstRecord object
```

```
  If (srcCon.Select (Nothing, 1, fldLst) <> 0) Then
    ' now map the fields of interest, in the order needed
    Call fetchLst.Map (fldLst, "ContactName, CompanyCity, CompanyState")
    Set cname = fetchLst.GetField (1)
    Set ccity = fetchLst.GetField (2)
    Set cstate = fetchLst.GetField (3)
    REM fetch a record from the result set
    srcCon.MapByName = True
```

```
    Print "Fetching ContactName, CompanyCity, and CompanyState fields"
```

```
    While (srcCon.Fetch (fetchLst, 1, 1) > 0)
      count = count + 1
      Print Cstr(count); Tab(3); cname.Text(0); Tab(28); _
        ccity.Text(0); Tab(42); cstate.Text(0)
```

```
    Wend
```

```
  End If
```

```
End Sub
```

Map method for LCFieldlist

Example Output

Fetching ContactName, CompanyCity, and CompanyState fields
1 Peter Pan Never Never Land
2 R. U. Happy Beagle WI
3 Issac Bernard Mathews New York NY

MapName method for LCFieldlist

This method maps fields with different names. Use this method for field mapping operations when the fieldnames are different in source and target. This allows you to change the order of fields and to exclude fields, as well.

Defined In

LCFieldlist

Syntax

Call *fldListRecod*.**MapName**(*baseFieldList*, *nameList*, *mapList*)

Parameters

<i>baseFieldList</i>	LCFieldlist. The fieldlist to map.
<i>nameList</i>	String. The names of the original fields.
<i>mapList</i>	String. The new names and mapping order for the fields.

Example

```
Option Public  
Uselsx "**Isxlc"
```

```
Sub Initialize
```

```
Dim session As New LCSession  
Dim srcCon As New LCConnection ("db2")  
Dim destCon As New LCConnection ("notes")  
Dim fetchLst As New LCFieldList  
Dim insertLst As New LCFieldList  
Dim count As Long
```

```
REM set the appropriate properties to connect to the data sources
```

```
srcCon.Database = "Gold"  
srcCon.Userid = "JDoe"  
srcCon.Password = "xyzyzy"  
srcCon.Metadata = "customer"
```

```
destCon.Server = "Rainbow"  
destCon.Database = "Gold"  
destCon.Metadata = "customer"
```

```
REM connect to the two data sources
```

```
srcCon.Connect  
destCon.Connect
```

```
srcCon.FieldNames = "ContactName, CompanyCity, CompanyState"
```

```
If (srcCon.Select (Nothing, 1, fetchLst) <> 0) Then
```

```
    ' map the result set from the SELECT with the desired names
```

```
    Call insertLst.MapName (fetchLst, _  
        "ContactName, CompanyCity, CompanyState", _  
        "Name, City, State")
```

```
    ' set the property MapbyName on the target.
```

```
    ' this is necessary if the fields in the form are
```

```
    ' (or might be) in differnt order from the mapping
```

```
    destCon.MapByName = True
```

```
While (srcCon.Fetch (fetchLst, 1, 1) > 0)
```

```
    count = count + destCon.Insert (insertLst, 1, 1)
```

```
Wend
```

```
Print "Transferred " & count & " records from DB2, to Notes"
```

```
Print "Field mapping from (DB2 column name to Notes Field name):"
```

```
Print "    ContactName --> Name"
```

```
Print "    CompanyCity --> City"
```

```
Print "    CompanyState --> State"
```

```
End If  
End Sub
```

Example Output

Transferred 3 records from DB2, to Notes

Field mapping from (DB2 column name to Notes Field name):

ContactName --> Name

CompanyCity --> City

CompanyState --> State

Merge method for LCFieldlist

This method merges two fieldlists, creating a new third mapping fieldlist. Use this method for field mapping operations when you either have a name fieldlist already built and/or you need to use fieldlist flags.

Defined In

LCFieldlist

Syntax

Call *fldListRecord.Merge*(*nameFieldList*, *dataFieldList*, *mergeFlags*)

Parameters

<i>nameFieldlist</i>	LCFieldlist. Source fieldlist containing the field names for the new fieldlist.
<i>dataFieldlist</i>	LCFieldlist. Source fieldlist containing the referenced data for the new fieldlist.
<i>mergeFlags</i>	<p>Long. By default, both fieldlists must have the same number of fields, and those fields are mapped by position (first to first, second to second, and so on). You can use <i>mergeFlags</i> to alter this default behavior, with zero or more of the following values, OR-ed together:</p> <p>LCMERGEF_MAP_NAME Match source and destination fields by field name instead of by position.</p> <p>LCMERGEF_DATA_LOSS Ignore fields in <i>DataFieldlist</i> that have no corresponding field in <i>NameFieldlist</i>.</p> <p>LCMERGEF_NAME_LOSS Ignore fields in <i>NameFieldlist</i> that have no corresponding field in <i>DataFieldlist</i>.</p>

LCMERGEF_FETCH

Ignore fields with the LCFIELDF_NO_FETCH flag set.

LCMERGEF_INSERT

Ignore fields with the LCFIELDF_NO_INSERT flag set.

LCMERGEF_UPDATE

Ignore fields with the LCFIELDF_NO_UPDATE flag set.

LCMERGEF_REMOVE

Ignore fields with the LCFIELDF_NO_REMOVE flag set.

LCMERGEF_CREATE

Ignore fields with the LCFIELDF_NO_CREATE flag set.

LCMERGEF_DROP

Ignore fields with the LCFIELDF_NO_DROP flag set.

LCMERGEF_KEY

Include fields with the LCFIELDF_KEY flag set.

Example

```
Option Public  
Uselsx "**Isxlc"
```

```
Sub Initialize
```

```
    Dim session As New LCSession  
    Dim srcCon As New LCConnection ("db2")  
    Dim destCon As New LCConnection ("notes")  
    Dim fldLst As New LCFieldlist  
    Dim fetchLst As New LCFieldlist  
    Dim insertLst As New LCFieldlist  
    Dim dataLst As New LCFieldlist  
    Dim nameLst As New LCFieldlist  
    Dim count As Long
```

```
    REM set the appropriate properties to connect to the data sources
```

```
    srcCon.Database = "Gold"  
    srcCon.Userid = "JDoe"  
    srcCon.Password = "xyzyz"  
    srcCon.Metadata = "customer"
```

```
    destCon.Server = "Rainbow"  
    destCon.Database = "Gold"  
    destCon.Metadata = "customer"
```

```
    REM connect to the two data sources
```

```
    srcCon.Connect  
    destCon.Connect
```

```
    REM we can perform a select and get the records with the fields we want in our fldLstRecord  
    object
```

```
    If (srcCon.Select (Nothing, 1, fldLst) <> 0) Then
```

```
        REM first we identify the data fields to fetch and order them
```

```
        Call dataLst.Append ("CONTACTNAME", LCTYPE_TEXT)
```

```
        Call dataLst.Append ("COMPANYSITY", LCTYPE_TEXT)
```

```
        Call dataLst.Append ("COMPANYSTATE", LCTYPE_TEXT)
```

```
        Call fetchLst.Merge (dataLst, fldLst, LCMERGEF_MAP_NAME Or LCMERGEF_DATA_LOSS)
```

```
        REM now we need to do a merge of the fields being fetched with the names of the fields being  
        stored
```

```
        Call nameLst.Append ("Name", LCTYPE_TEXT)
```

```
        Call nameLst.Append ("City", LCTYPE_TEXT)
```

```
        Call nameLst.Append ("State", LCTYPE_TEXT)
```

```
        Call insertLst.Merge (nameLst, fetchLst, 0)
```

```
    REM set the property Map by Name on both data sources
```

```
    srcCon.MapByName = True  
    destCon.MapByName = True
```

```
REM fetch a record from the result set
While (srcCon.Fetch (fetchLst, 1, 1) > 0)
    REM now insert the record into the target and fetch the next, looping until all records have
    been inserted
    count = count + destCon.Insert (insertLst, 1, 1)
Wend
Print "Transferred " & count & " records from DB2, to Notes"
Print "Field mapping from (DB2 column name to Notes Field name):"
Print "   ContactName  --> Name"
Print "   CompanyCity  --> City"
Print "   CompanyState --> State"
End If
End Sub
```

Example Output

```
Transferred 3 records from DB2, to Notes
Field mapping from (DB2 column name to Notes Field name):
ContactName  --> Name
CompanyCity  --> City
CompanyState --> State
```

MergeVirtual method for LCFieldlist

This method merges two fieldlists, creating new mapping and virtual fieldlists.

NOTE: This method is provided for backward compatibility. We recommend that you use either the Map or MapName methods for merging fieldlists and creating new mappings.

Defined In

LCFieldlist

Syntax

Call *fldList.MergeVirtual(nameFieldList, dataFieldList, MergeFlags, virtualCode, virtualFieldList)*

Parameters

<i>nameFieldlist</i>	LCFieldlist. Source fieldlist containing the field list of names for the new fieldlist.
<i>dataFieldlist</i>	LCFieldlist. Source fieldlist containing the referenced data for the new fieldlist.
<i>mergeFlags</i>	<p>Long. By default, both fieldlists must have the same number of fields, and those fields are mapped by position (first to first, second to second, an so on). Use <i>MergeFlags</i> to alter this default behavior. Zero or more of the following values, OR-ed together:</p> <p>LCMERGEF_MAP_NAME Match source and destination fields by field name instead of by position. (See Comment.)</p> <p>LCMERGEF_DATA_LOSS Ignore fields in <i>DataFieldlist</i> that have no corresponding field in <i>NameFieldlist</i>.</p> <p>LCMERGEF_NAME_LOSS Ignore fields in <i>NameFieldlist</i> that have no corresponding field in <i>DataFieldlist</i>.</p>

LCMERGEF_FETCH

Ignore fields with the LCFIELDF_NO_FETCH flag set.

LCMERGEF_INSERT

Ignore fields with the LCFIELDF_NO_INSERT flag set.

LCMERGEF_UPDATE

Ignore fields with the LCFIELDF_NO_UPDATE flag set.

LCMERGEF_REMOVE

Ignore fields with the LCFIELDF_NO_REMOVE flag set.

LCMERGEF_CREATE

Ignore fields with the LCFIELDF_NO_CREATE flag set.

LCMERGEF_DROP

Ignore fields with the LCFIELDF_NO_DROP flag set.

LCMERGEF_KEY

Include fields with the LCFIELDF_KEY flag set.

virtualCode

Long. Connect or Connector code to separate fields with a matching virtual code into *virtualFieldlist*.

virtualFieldlist

LCFieldlist. New fieldlist, containing fields with virtual codes matching *virtualCode*.

Remove method for LCFieldlist

This method removes an existing field from a fieldlist.

Defined In

LCFieldlist

Syntax

Call *fldLstRecord.Remove* (*index*)

Parameters

<i>index</i>	Long. The index position of the field to be removed from the fieldlist.
--------------	---

Example

```
Option Public
Uselsx "*lsxlc"
```

```
Sub Initialize
```

```
  Dim connect As New LCConnection ("db2")
  Dim FldLst As New LCFieldlist
  Dim field As LCField
  Dim text As String
```

```
  REM this section assigns the appropriate properties to connect to DB2
```

```
  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "xyzyzy"
  connect.Metadata = "customer"
```

```
  REM connect to DB2
```

```
  connect.Connect
```

```
  REM now perform the catalog action - in this case for fields
```

```
  If (connect.Select (Nothing, 1, FldLst) = 0) Then
```

```
    Print "The customer table was not found."
```

```
  Else
```

```
    REM fetch the field names
```

```
    Call FldLst.Remove (1)
```

```
    text = ""
```

```
    Forall fieldname In FldLst.Names
```

```
      If (text = "") Then text = fieldname Else text = text + ", " + fieldname
```

```
    End Forall
```

```
    Print "The new list of columns in the " & connect.Metadata & " table are: " & text
```

```
  End If
```

```
End Sub
```

Example Output

The new list of columns in the 'customer' table are: CONTACTNAME, COMPANYNAME, COMPANYADDRESS, COMPANYCITY, COMPANYSSTATE, COMPANYPHONE

Replace method for LCFieldlist

This method replaces a field within a fieldlist.

Defined In

LCFieldlist

Syntax

Call *fldLstRecord*.**Replace** (*index*, *fieldName*, *dataType*)

Parameters

<i>index</i>	Long. The index position of the field to replace.
<i>fieldName</i>	String. The name of the new field.
<i>dataType</i>	Long. The datatype to assign to the new field.

Example

```
Option Public
Uselsx "*lsxlc"
```

```
Sub Initialize
```

```
  Dim connect As New LCConnection ("db2")
  Dim FldLst As New LCFieldlist
  Dim field As LCField
  Dim text As String
```

```
  REM this section assigns the appropriate properties to connect to DB2
```

```
  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "xyzyzy"
  connect.Metadata = "customer"
```

```
  REM connect to DB2
```

```
  connect.Connect
```

```
  REM now perform the catalog action - in this case for fields
```

```
  If (connect.Select (Nothing, 1, FldLst) = 0) Then
```

```
    Print "The customer table was not found."
```

```
  Else
```

```
    REM fetch the field names
```

```
    Call FldLst.Replace (2, "pinky", LCTYPE_TEXT)
```

```
    text = ""
```

```
    Forall fieldname In FldLst.Names
```

```
      If (text = "") Then text = fieldname Else text = text + ", " + fieldname
```

```
    End Forall
```

```
    Print "The new list of columns in the " & connect.Metadata & " table are: " & text
```

```
  End If
```

```
End Sub
```

Example Output

The new list of columns in the 'customer' table are: ACCOUNTMANAGER, pinky, COMPANYNAME, COMPANYADDRESS, COMPANYCITY, COMPANYSTATE, COMPANYPHONE

SetName method for LCFieldlist

This method changes the name of a field within a fieldlist.

Defined In

LCFieldlist

Syntax

Call *fldLstRecord*.**SetName**(*index*, *fieldName*)

Parameters

<i>index</i>	Long. The index position of the field.
<i>fieldName</i>	String. The new name to give to this field.

Example

```
Option Public
Uselsx "*lsxlc"
```

```
Sub Initialize
```

```
  Dim connect As New LCCConnection ("db2")
  Dim FldLst As New LCFieldlist
  Dim field As LCField
  Dim text As String
```

```
  REM this section assigns the appropriate properties to connect to DB2
```

```
  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "xyzyzy"
  connect.Metadata = "customer"
```

```
  REM connect to DB2
```

```
  connect.Connect
```

```
  REM now perform the catalog action - in this case for fields
```

```
  If (connect.Select (Nothing, 1, FldLst) = 0) Then
```

```
    Print "The customer table was not found."
```

```
  Else
```

```
    REM fetch the field names
```

```
    Call FldLst.SetName (1, "pinky")
```

```
    text = ""
```

```
    Forall fieldname In FldLst.Names
```

```
      If (text = "") Then text = fieldname Else text = text + ", " + fieldname
```

```
    End Forall
```

```
    Print "The new list of columns in the " & connect.Metadata & " table are: " & text
```

```
  End If
```

```
End Sub
```

Example Output

The new list of columns in the 'customer' table are: pinky, CONTACTNAME, COMPANYNAME, COMPANYADDRESS, COMPANYCITY, COMPANYSTATE, COMPANYPHONE

Chapter 7

LCNumeric Class

This chapter provides information about the Lotus Connectors LCNumeric class methods and properties.

Overview

The LCNumeric class represents a numeric value containing a precision, scale, and variable number of digits. Numeric values have a specific precision and scale, and can accommodate very high-precision numbers. By default, a new numeric submitted to any other numeric method is initialized to a precision of 88 and a scale of 44. Note that during any numeric overflow, the maximum or minimum valid numeric value is assigned in addition to the error occurring.

For a numeric to be valid, it must have valid values for precision (number of digits) and scale (number of decimal places). The precision and scale of a numeric may only be set when it is first created. If precision and scale are not valid for a numeric submitted as parameters when creating a numeric, then an error is generated. If the invalid numeric has been zeroed, then it will be automatically initialized to a numeric with the maximum precision and a scale of precision/2 (44).

Type NUMERIC format and values

Precision (LCMAX_NUMERIC_PREC) = 88
Minimum Scale (LCMIN_NUMERIC_SCALE) = -127
Maximum Scale (LCMAX_NUMERIC_SCALE) = 127
Minimum Positive Value (LCMIN_NUMERIC_VALUE) = 1.0e ⁻¹²⁷
Maximum Positive Value (LCMAX_NUMERIC_VALUE) = 9.99... e ¹²⁶

LCNumeric Properties

LCNumeric.Precision	Long [Read-Only]. Precision and Scale are set when the LCNumeric object is first constructed. See the New method for LCNumeric.
LCNumeric.Scale	Long [Read-Only]. Precision and Scale are set when the LCNumeric object is first constructed. See the New method for LCNumeric.
LCNumeric.Text	String representation.
LCNumeric.Value	Double - value conversion between Lotus Connectors and LotusScript double.

LCNumeric Class Methods Summary

LCNumeric.Add	Adds two numeric values, producing the sum.
LCNumeric.Compare	Compares two numeric values returning a value indicating the relationship between them.
LCNumeric.Copy	Makes a copy of a numeric value.
LCNumeric.Subtract	Subtracts one numeric value from another, producing the result.

New method for LCNumeric

Constructor for new LCNumeric class object.

Defined In

LCNumeric

Syntax

Dim variableName As New LCNumeric(long1, long2)

Parameters

<i>long1</i>	Long. Optional. Precision for this object, from 0 - 88. Default is LCMAX_NUMERIC_PREC (88).
<i>long2</i>	Long. Optional. Scale for this object, from -127 - +127. Default is LCMAX_NUMERIC_PREC / 2 (44).

Add method for LCNumeric

This method adds the values of two LCNumeric objects.

Defined In

LCNumeric

Syntax

Call *numericTotal*.**Add**(*numeric1*, *numeric2*)

Parameters

<i>numeric1</i>	The first of two values being added.
<i>numeric2</i>	The second of two values being added.

Add method for LCNumeric

Example

```
Option Public  
Uselsx "*Isxlc"
```

```
Sub Initialize  
    Dim num1 As New LCNumeric  
    Dim num2 As New LCNumeric  
    Dim sum As New LCNumeric  
  
    num1.Value = 12345.6789  
    num2.Value = 12345.6789  
    Call sum.Add (Num1, Num2)  
    Print "The sum of the two numbers is " & sum.Text  
End Sub
```

Example Output

The sum of the two numbers is 24691.3578

Compare method for LCNumeric

This method compares two LCNumeric objects.

Defined In

LCNumeric

Syntax

result = *numericFirst*.**Compare**(*numericSecond*)

Parameters

numericSecond The base value to which to compare another value.

Return Value

result Result of the comparison:

Result > 0 (positive): *numericFirst* is greater than *numericSecond*.

Result < 0 (negative): *numericFirst* is less than *numericSecond*.

Result = 0: *numericFirst* is equal to *numericSecond*.

Example

```
Option Public
Uselsx "*Isxlc"

Sub Initialize
    Dim num1 As New LCNumeric
    Dim num2 As New LCNumeric

    num1.Value = 123.456789
    num2.Value = 123

    If (num1.Compare (num2) = 0) Then
        Print "The first number is the same as the second."
    ElseIf (num1.Compare (num2) > 0) Then
        Print "The first number is greater than the second."
    Else
        Print "The first number is less than the second."
    End If
End Sub
```

Example Output

The first number is greater than the second.

Copy method for LCNumeric

This method makes a copy of one LCNumeric, creating a new LCNumeric object.

Defined In

LCNumeric

Syntax

Set *newNumeric* = *srcFirst*.**Copy**

Parameters

srcNumeric LCNumeric. The LCNumeric object that you want to copy.

Return Value

newNumeric LCNumeric. The copy of the *srcNumeric* object.

Copy method for LCNumeric

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize  
    Dim num1 As New LCNumeric  
    Dim num2 As LCNumeric  
  
    num1.Value = 12345.6789  
    Set num2 = num1.Copy  
    Print "The copy has a value of " & num2.Text  
End Sub
```

Example Output

The copy has a value of 12345.6789

Subtract method for LCNumeric

This method subtracts the value of one LCNumeric object from the value of another LCNumeric object.

Defined In

LCNumeric

Syntax

Call *numericThird*.**Subtract**(*numericFirst*, *numericSecond*)

Parameters

numericFirst The initial value from which to subtract *numericSecond*.

numericSecond The value to subtract from *numericFirst*.

Example

```
Option Public  
Uselsx "**Isxlc"
```

```
Sub Initialize  
    Dim num1 As New LCNumeric  
    Dim num2 As New LCNumeric  
    Dim diff As New LCNumeric  
  
    num1.Value = 98765.4321  
    num2.Value = 12345.6789  
    Call diff.Subtract (Num1, Num2)  
    Print "The difference of the two numbers is " & diff.Text  
End Sub
```

Example Output

The difference of the two numbers is 86419.7532

Chapter 8

LCSession Class

This chapter provides information about Lotus Connectors LCSession class methods and properties.

Overview

The LCSession class represents the Lotus Connectors environment of the current script, providing access to the available connectors and metaconnectors, as well as the current error status. The status property and methods are useful when writing error handling code and reporting errors as text messages.

Properties

LCSession.Status	Status of an LCSession. Zero, or LCSUCCESS, indicates no error. A non-zero value (represented by an LCFAIL_XXX constant) indicates an error state.
------------------	--

LCSession Class Methods Summary

LCSession.ClearStatus	Clear the Session status, putting the Session in a non-error state.
LCSession.GetStatus	Obtain the current Session status.
LCSession.GetStatusText	Obtain the error text string corresponding to a status code.
LCSession.ListConnector	List through all installed connectors available for LotusScript Extension for Lotus Connectors.
LCSession.ListMetaConnector	List through all installed metaconnectors available for a LotusScript Extension for Lotus Connectors installation.
LCSession.LookupConnector	Determines if a specified connector is available.

Overview

LCSession.LookupMetaConnector	Determines if a specified metaconnector is available.
LCSession.Sleep	Suspends script execution for a specified period of time.

New method for LCSession

Constructor for LCSession object.

Multiple LCSession objects may be created within a script. All LCSession objects within a single script execution share the same status information.

Defined In

LCSession

Syntax

Dim variableName As New LCSession

ClearStatus method for LCSession

This method resets, or clears, the LCSession status after an error.

Defined In

LCSession

Syntax

thisSession.**ClearStatus**()

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim session As New LCSession  
  
    REM Ignore errors  
    On Error Resume Next  
    REM purposely generate a Lotus Connector error  
    session.ListConnector (23)  
    Print "The current status code is #" & Cstr (session.Status)  
    session.ClearStatus  
    Print "The new status code is #" & Cstr (session.Status)  
End Sub
```

Example Output

```
The current status code is #12292  
The new status code is #0
```

GetStatus method for LCSession

This method retrieves the current status value of a session.

Defined In

LCSession

Syntax

Call *thisSession*.**GetStatus** (*statusText*, *externalCode*, *externalText*)

Parameters

<i>statusText</i>	String. Optional. Lotus Connector status message.
<i>externalCode</i>	Long. Optional. Any external error codes. Long.
<i>externalText</i>	String. Optional. Any external message text associated with an external code.

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
    On Error Goto handler
    Dim session As New LCSession
    Dim src As New LCConnection ("db2")
```

```
    src.Database = "Gold"
    src.UserID = "JDoe"
    REM deliberate bad password
    src.Password = "xyzzz"
    src.Connect
    Print "Connected to DB2."
End
```

handler:

```
If (session.Status <> LCSUCCESS) Then
    Dim text As String
    Dim extcode As Long
    Dim exttext As String

    Call session.GetStatus (text, extcode, exttext)
    If (session.Status = LCFAIL_EXTERNAL)Then
        Print "DB2 message: " & exttext & " code #" & Cstr(extcode)
    Else
        Print "Connector message: " & text
    End If
Else
    Print Error$
End If
End
End Sub
```

Example Output

DB2 message: [IBM][CLI Driver] SQL1403N The username and/or password supplied is incorrect. SQLSTATE=08004 code #-1403

GetStatusText method for LCSession

This method returns the message text corresponding to an LCStatus code.

Defined In

LCSession

Syntax

message = *thisSession*.GetStatusText(*errorCode*)

Parameters

errorCode

Long. Optional. The error code for which to return the message text. The default is the current status code.

Return Values

message

A text string representation of the status.

Example

```
Option Public
Uselsx "**lsxlc"

Sub Initialize
    Dim session As New LCSession

    REM Ignore errors
    On Error Resume Next
    REM purposely generate a Lotus Connector error
    session.ListConnector (23)
    If (session.Status <> LCSUCCESS) Then
        Print session.GetStatusText
    Else
        Print "No error exists."
    End If
End Sub
```

Example Output

Error: Invalid List direction

ListConnector method for LCSession

This method lists through all valid connectors for a Lotus Extension for Lotus Connectors installation.

The identifyFlagList and identifyNameList parameters provide the ability to obtain information from a Lotus Connector about its supported functionality and naming used by the backend system.

Defined In

LCSession

Syntax

Call *thisSession*.**ListConnector**(*List*, *connectorName*, *connectorCode*, *identifyFlagList*, *identifyNameList*)

Parameters

<i>list</i>	Long. Constant indicating whether to return the first or next Connector property. LCLIST_FIRST Return the first property in the property list. LCLIST_NEXT Return the next property (or the first property if this is the first call to this function for this Connection).
<i>connectorName</i>	String. Optional. String representation of the connector name.
<i>connectorCode</i>	String. Optional. The assigned Connector code.
<i>identifyFlagList</i>	Stream. Optional. <i>IdentifyFlagList</i> is set to a stream of format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are

indicated by the index in the number list. Use the following constants to represent a particular set of flags (default is Nothing):

LCIDFLAG_INDEX_CONNECTOR

Connector flags

LCIDENTIFYF_XXX

LCIDFLAG_INDEX_ACTION

Support actions for LCCConnection.Action method

LCACTIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_CATALOG

Support objects for LCCConnection.Catalog method

LCOBJIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_CREATE

Support objects for LCCConnection.Catalog method

LCOBJIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_DROP

Support objects for LCCConnection.Catalog method

LCOBJIDENTF_XXX

The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags OR-ed together:

Supported Connector flags:

LCIDENTIFYF_SINGLE_THREAD

Connector is not thread safe (NOTE: The LSX LC will properly serialize access to this connector to avoid threading problems.

LCIDENTIFYF_ARRAY_READ

Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported.

LCIDENTIFYF_ARRAY_WRITE

Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported.

LCIDENTIFYF_SINGLE_METADATA

All data is represented by a single metadata (for example, the File connector has only one 'metadata' description)

LCIDENTIFYF_WRITEBACK

ListConnector method for LCSession

Writeback functionality is available

LCIDENTIFYF_SCROLLING

Scrolling result sets are available (currently not supported by any connectors).

LCIDENTIFYF_MULTI_VALUE

Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the backend.

LCIDENTIFYF_MULTI_DIMENSION

Multi-dimensional result sets are supported (nested fieldlists).

LCIDENTIFYF_SQL

SQL is the backend-supported syntax.

LCIDENTIFYF_SRVDB_CAT_CONNECT

Database connection is required for server and/or database browsing.

LCIDENTIFYF_DISABLE_WRITEBACK

(Metaconnector only) The use of this metaconnector does not allow writeback result sets.

Supported action flags:

LCACTIDENTF_RESET

Reset action is supported

LCACTIDENTF_TRUNCATE

Truncate action is supported

LCACTIDENTF_COMMIT

Commit action is supported

LCACTIDENTF_ROLLBACK

Rollback action is supported

LCACTIDENTF_CLEAR

Clear action is supported

LCACTIDENTF_WAIT

Wait action is supported

Supported object flags:

LCOBJIDENTF_SERVER

The method supports server objects

LCOBJIDENTF_DATABASE

The method supports database objects

LCOBJIDENTF_METADATA

The method supports metadata objects

LCOBJIDENTF_PROCEDURE

The method supports procedure objects

LCOBJIDENTF_INDEX

The method supports index objects

LCOBJIDENTF_FIELD

The method supports field objects

LCOBJIDENTF_PARAMETER

The method supports parameter objects

LCOBJIDENTF_ALT_METADATA

The method supports alternate metadata objects

LCOBJIDENTF_ALT_FIELD

The method supports alternate metadata field objects

identifyNameList

Stream. Optional. *IdentifyNameList* is set to a stream of format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's backend system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing):

LCIDNAME_INDEX_SERVER

Name for server objects in this backend system.

LCIDNAME_INDEX_DATABASE

Name for database objects in this backend system.

LCIDNAME_INDEX_USERID

Name for user ID objects in this backend system.

ListConnector method for LCSession

LCIDNAME_INDEX_PASSWORD

Name for password objects in this backend system.

LCIDNAME_INDEX_METADATA

Name for metadata objects in this backend system.

LCIDNAME_INDEX_PROCEDURE

Name for procedure objects in this backend system.

LCIDNAME_INDEX_INDEX

Name for index objects in this backend system.

LCIDNAME_INDEX_FIELD

Name for metadata fields in this backend system.

LCIDNAME_INDEX_PARAMETER

Name for procedure parameters in this backend system.

LCIDNAME_INDEX_ALT_METADATA

Name for alternate metadata objects in this backend system.

LCIDNAME_INDEX_ALT_FIELD

Name for alternate metadata fields in this backend.

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim session As New LCSession  
    Dim conName As String  
    Dim text As String  
  
    REM list the connectors available  
    REM the parameters for connector code, identity flags, and  
    REM identity names are optional and omitted in this example  
    Call session.ListConnector(LCLIST_FIRST, conName)  
    text = conName  
    While session.ListConnector(LCLIST_NEXT, conName)  
        text = text + ", " + conName  
    Wend  
    Print "The usable Connectors are " & text  
End Sub
```

Example Output

The usable Connectors are db2, notes, odbc2, oracle, sybase

ListMetaConnector method for LCSession

This method lists all valid metaconnectors for a Lotus Connectors installation.

Defined In

LCSession

Syntax

Call thisSession.**ListMetaConnector**(*list*, *metaconnectorName*, *connectorCode*, *identifyFlagList*, *identifyNameList*)

Parameters

<i>list</i>	Long. Constant indicating whether to return the first or next MetaConnector property. LCLIST_FIRST Return the first property in the property list. LCLIST_NEXT Return the next property (or the first property if this is the first call to this function for this Connection).
<i>metaconnectorName</i>	String. Optional. The name of the metaconnector.
<i>connectorCode</i>	String. Optional. The code for this metaconnector.
<i>identifyFlagList</i>	LCStream. Optional. <i>IdentifyFlagList</i> is set to a stream of format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are indicated by the index in the number list. Use the following constants to represent a particular set of flags: LCIDFLAG_INDEX_CONNECTOR Connector flags LCIDENTIFYF_XXX

LCIDFLAG_INDEX_ACTION

Support actions for LCConnection.Action method
LCACTIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_CATALOG

Support objects for LCConnection.Catalog method
LCOBJIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_CREATE

Support objects for LCConnection.Catalog method
LCOBJIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_DROP

Support objects for LCConnection.Catalog method
LCOBJIDENTF_XXX

The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags OR-ed together:

Supported Connector flags:

LCIDENTIFYF_SINGLE_THREAD

Connector is not thread safe (NOTE: The LSX LC will properly serialize access to this connector to avoid threading problems).

LCIDENTIFYF_ARRAY_READ

Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported.

LCIDENTIFYF_ARRAY_WRITE

Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported.

LCIDENTIFYF_SINGLE_METADATA

All data is represented by a single metadata (for example, the File connector has only one 'metadata' description)

LCIDENTIFYF_WRITEBACK

Writeback functionality is available

LCIDENTIFYF_SCROLLING

Scrolling result sets are available (currently not supported by any connectors).

LCIDENTIFYF_MULTI_VALUE

ListMetaConnector method for LCSession

Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the backend.

LCIDENTIFYF_MULTI_DIMENSION

Multi-dimensional result sets are supported (nested fieldlists).

LCIDENTIFYF_SQL

SQL is the backend-supported syntax.

LCIDENTIFYF_SRVDB_CAT_CONNECT

Database connection is required for server and/or database browsing.

LCIDENTIFYF_DISABLE_WRITEBACK

(Metaconnector only) The use of this metaconnector does not allow writeback result sets.

Supported action flags:

LCACTIDENTF_RESET

Reset action is supported

LCACTIDENTF_TRUNCATE

Truncate action is supported

LCACTIDENTF_COMMIT

Commit action is supported

LCACTIDENTF_ROLLBACK

Rollback action is supported

LCACTIDENTF_CLEAR

Clear action is supported

LCACTIDENTF_WAIT

Wait action is supported

Supported object flags:

LCOBJIDENTF_SERVER

The method supports server objects

LCOBJIDENTF_DATABASE

The method supports database objects

LCOBJIDENTF_METADATA

The method supports metadata objects

LCOBJIDENTF_PROCEDURE

The method supports procedure objects

LCOBJIDENTF_INDEX

The method supports index objects

LCOBJIDENTF_FIELD

The method supports field objects

LCOBJIDENTF_PARAMETER

The method supports parameter objects

LCOBJIDENTF_ALT_METADATA

The method supports alternate metadata objects

LCOBJIDENTF_ALT_FIELD

The method supports alternate metadata field objects

identifyNameList

LCStream. Optional. *IdentifyNameList* is set to a stream of format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's backend system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing):

LCIDNAME_INDEX_SERVER

Name for server objects in this backend system.

LCIDNAME_INDEX_DATABASE

Name for database objects in this backend system.

LCIDNAME_INDEX_USERID

Name for user ID objects in this backend system.

LCIDNAME_INDEX_PASSWORD

Name for password objects in this backend system.

LCIDNAME_INDEX_METADATA

Name for metadata objects in this backend system.

ListMetaConnector method for LCSession

LCIDNAME_INDEX_PROCEDURE

Name for procedure objects in this backend system.

LCIDNAME_INDEX_INDEX

Name for index objects in this backend system.

LCIDNAME_INDEX_FIELD

Name for metadata fields in this backend system.

LCIDNAME_INDEX_PARAMETER

Name for procedure parameters in this backend system.

LCIDNAME_INDEX_ALT_METADATA

Name for alternate metadata objects in this backend system.

LCIDNAME_INDEX_ALT_FIELD

Name for alternate metadata fields in this backend.

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
  Dim session As New LCSession
  Dim conName As String
  Dim text As String

  REM list the connectors available
  REM the parameters for connector code, identity flags, and
  REM identity names are optional and omitted in this example
  Call session.ListMetaConnector(LCLIST_FIRST, conName)
  text = conName
  While session.ListMetaConnector(LCLIST_NEXT, conName)
    text = text + ", " + conName
  Wend
  Print "The usable MetaConnector(s) are " & text
End Sub
```

Example Output

The usable MetaConnector(s) are collexp, order

LookupConnector method for LCSession

This method looks up a connector name to determine if it exists and returns information about the connector.

Defined In

LCSession

Syntax

Call *thisSession.LookupConnector*(*connectorName*, *connectorCode*, *identifyFlagList*, *identifyNameList*)

Parameters

<i>connectorName</i>	String. The name of the Connector to look up, for example, "oracle."
<i>connectorCode</i>	Long. Optional. The Connector code for this Connector.
<i>identifyFlagList</i>	<p>LCStream. Optional. <i>IdentifyFlagList</i> is set to a stream of format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are indicated by the index in the number list. Use the following constants to represent a particular set of flags:</p> <p>LCIDFLAG_INDEX_CONNECTOR Connector flags LCIDENTIFYF_XXX</p> <p>LCIDFLAG_INDEX_ACTION Support actions for LCConnection.Action method LCACTIDENTF_XXX</p> <p>LCIDFLAG_INDEX_OBJECT_CATALOG Support objects for LCConnection.Catalog method LCOBJIDENTF_XXX</p>

LCIDFLAG_INDEX_OBJECT_CREATE

Support objects for LCConnection.Catalog method

LCOBJIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_DROP

Support objects for LCConnection.Catalog method

LCOBJIDENTF_XXX

The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags OR-ed together:

Supported Connector flags:

LCIDENTIFYF_SINGLE_THREAD

Connector is not thread safe (NOTE: The LSX LC will properly serialize access to this connector to avoid threading problems).

LCIDENTIFYF_ARRAY_READ

Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported.

LCIDENTIFYF_ARRAY_WRITE

Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported.

LCIDENTIFYF_SINGLE_METADATA

All data is represented by a single metadata (for example, the File connector has only one 'metadata' description)

LCIDENTIFYF_WRITEBACK

Writeback functionality is available

LCIDENTIFYF_SCROLLING

Scrolling result sets are available (currently not supported by any connectors).

LCIDENTIFYF_MULTI_VALUE

Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the backend.

LCIDENTIFYF_MULTI_DIMENSION

Multi-dimensional result sets are supported (nested fieldlists).

LCIDENTIFYF_SQL

SQL is the backend-supported syntax.

LookupConnector method for LCSession

LCIDENTIFYF_SRVDB_CAT_CONNECT

Database connection is required for server and/or database browsing.

LCIDENTIFYF_DISABLE_WRITEBACK

(Metaconnector only) The use of this metaconnector does not allow writeback result sets.

Supported action flags:

LCACTIDENTF_RESET

Reset action is supported

LCACTIDENTF_TRUNCATE

Truncate action is supported

LCACTIDENTF_COMMIT

Commit action is supported

LCACTIDENTF_ROLLBACK

Rollback action is supported

LCACTIDENTF_CLEAR

Clear action is supported

LCACTIDENTF_WAIT

Wait action is supported

Supported object flags:

LCOBJIDENTF_SERVER

The method supports server objects

LCOBJIDENTF_DATABASE

The method supports database objects

LCOBJIDENTF_METADATA

The method supports metadata objects

LCOBJIDENTF_PROCEDURE

The method supports procedure objects

LCOBJIDENTF_INDEX

The method supports index objects

LCOBJIDENTF_FIELD

The method supports field objects

LCOBJIDENTF_PARAMETER

The method supports parameter objects

LCOBJIDENTF_ALT_METADATA

The method supports alternate metadata objects

LCOBJIDENTF_ALT_FIELD

The method supports alternate metadata field objects

identifyNameList

LCStream. Optional. *IdentifyNameList* is set to a stream of format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's backend system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing):

LCIDNAME_INDEX_SERVER

Name for server objects in this backend system.

LCIDNAME_INDEX_DATABASE

Name for database objects in this backend system.

LCIDNAME_INDEX_USERID

Name for user ID objects in this backend system.

LCIDNAME_INDEX_PASSWORD

Name for password objects in this backend system.

LCIDNAME_INDEX_METADATA

Name for metadata objects in this backend system.

LCIDNAME_INDEX_PROCEDURE

Name for procedure objects in this backend system.

LCIDNAME_INDEX_INDEX

Name for index objects in this backend system.

LCIDNAME_INDEX_FIELD

Name for metadata fields in this backend system.

LookupConnector method for LCSession

LCIDNAME_INDEX_PARAMETER

Name for procedure parameters in this backend system.

LCIDNAME_INDEX_ALT_METADATA

Name for alternate metadata objects in this backend system.

LCIDNAME_INDEX_ALT_FIELD

Name for alternate metadata fields in this backend.

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim session As New LCSession  
  
    REM the optional parameters have been omitted in this example  
    If (session.LookupConnector ("db2")) Then  
        Print "DB2 connectivity installed."  
    Else  
        Print "DB2 connectivity is not installed."  
    End If  
End Sub
```

Example Output

DB2 connectivity installed.

LookupMetaConnector method for LCSession

This method looks up a metaconnector name to determine if it exists and returns information about the metaconnector.

Defined In

LCSession

Syntax

Call *thisSession.LookupMetaConnector*(*metaconnectorName*, *connectorCode*, *tokenBase*, *identifyFlagList*, *identifyNameList*)

Parameters

<i>metaconnectorName</i>	String. The name of the metaconnector to look up.
<i>connectorCode</i>	Long. Optional. The Connector code of the metaconnector.
<i>tokenBase</i>	Long. Optional. The tokenbase for this metaconnector.
<i>identifyFlagList</i>	<p>LCStream. Optional. <i>IdentifyFlagList</i> is set to a stream of format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are indicated by the index in the number list. Use the following constants to represent a particular set of flags:</p> <p>LCIDFLAG_INDEX_CONNECTOR Connector flags LCIDENTIFYF_XXX</p> <p>LCIDFLAG_INDEX_ACTION Support actions for LCConnection.Action method LCACTIDENTF_XXX</p> <p>LCIDFLAG_INDEX_OBJECT_CATALOG Support objects for LCConnection.Catalog method</p>

LCOBJIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_CREATE

Support objects for LCConnection.Catalog method

LCOBJIDENTF_XXX

LCIDFLAG_INDEX_OBJECT_DROP

Support objects for LCConnection.Catalog method

LCOBJIDENTF_XXX

The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags OR-ed together:

Supported Connector flags:

LCIDENTIFYF_SINGLE_THREAD

Connector is not thread safe (NOTE: The LSX LC will properly serialize access to this connector to avoid threading problems).

LCIDENTIFYF_ARRAY_READ

Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported.

LCIDENTIFYF_ARRAY_WRITE

Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported.

LCIDENTIFYF_SINGLE_METADATA

All data is represented by a single metadata (for example, the File connector has only one 'metadata' description)

LCIDENTIFYF_WRITEBACK

Writeback functionality is available

LCIDENTIFYF_SCROLLING

Scrolling result sets are available (currently not supported by any connectors).

LCIDENTIFYF_MULTI_VALUE

Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the backend.

LCIDENTIFYF_MULTI_DIMENSION

Multi-dimensional result sets are supported (nested fieldlists).

LookupMetaConnector method for LCSession

LCIDENTIFYF_SQL

SQL is the backend-supported syntax.

LCIDENTIFYF_SRVDB_CAT_CONNECT

Database connection is required for server and/or database browsing.

LCIDENTIFYF_DISABLE_WRITEBACK

(Metaconnector only) The use of this metaconnector does not allow writeback result sets.

Supported action flags:

LCACTIDENTF_RESET

Reset action is supported

LCACTIDENTF_TRUNCATE

Truncate action is supported

LCACTIDENTF_COMMIT

Commit action is supported

LCACTIDENTF_ROLLBACK

Rollback action is supported

LCACTIDENTF_CLEAR

Clear action is supported

LCACTIDENTF_WAIT

Wait action is supported

Supported object flags:

LCOBJIDENTF_SERVER

The method supports server objects

LCOBJIDENTF_DATABASE

The method supports database objects

LCOBJIDENTF_METADATA

The method supports metadata objects

LCOBJIDENTF_PROCEDURE

The method supports procedure objects

LCOBJIDENTF_INDEX

The method supports index objects

LCOBJIDENTF_FIELD

The method supports field objects

LCOBJIDENTF_PARAMETER

The method supports parameter objects

LCOBJIDENTF_ALT_METADATA

The method supports alternate metadata objects

LCOBJIDENTF_ALT_FIELD

The method supports alternate metadata field objects

identifyNameList

LCStream. Optional. *IdentifyNameList* is set to a stream of format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's backend system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing):

LCIDNAME_INDEX_SERVER

Name for server objects in this backend system.

LCIDNAME_INDEX_DATABASE

Name for database objects in this backend system.

LCIDNAME_INDEX_USERID

Name for user ID objects in this backend system.

LCIDNAME_INDEX_PASSWORD

Name for password objects in this backend system.

LCIDNAME_INDEX_METADATA

Name for metadata objects in this backend system.

LCIDNAME_INDEX_PROCEDURE

Name for procedure objects in this backend system.

LCIDNAME_INDEX_INDEX

Name for index objects in this backend system.

LookupMetaConnector method for LCSession

LCIDNAME_INDEX_FIELD

Name for metadata fields in this backend system.

LCIDNAME_INDEX_PARAMETER

Name for procedure parameters in this backend system.

LCIDNAME_INDEX_ALT_METADATA

Name for alternate metadata objects in this backend system.

LCIDNAME_INDEX_ALT_FIELD

Name for alternate metadata fields in this backend.

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim session As New LCSession  
  
    REM the optional parameters have been omitted in this example  
    If (session.LookupMetaConnector ("order")) Then  
        Print "The 'order' meta connector is present."  
    Else  
        Print "The 'order' meta connector is not present."  
    End If  
End Sub
```

Example Output

The 'order' meta connector is present.

Sleep method for LCSession

This method forces a script execution to sleep for the specified length of time.

Defined In

LCSession

Syntax

Call *thisSession.Sleep(milliSeconds)*

Parameters

milliSeconds Long. Number of milliseconds to sleep.

Example

```
Option Public  
Uselx "xlxl"
```

```
Sub Initialize  
    Dim session As New LCSession
```

```
    Print "The Time is " Cstr(Now)  
    session.Sleep (5000)  
    Print "The Time is " Cstr(Now)  
End Sub
```

Example Output

```
The Time is 9/8/98 5:23:32 PM  
The Time is 9/8/98 5:23:37 PM
```

Chapter 9

LCStream Class

This chapter provides information about the Lotus Connector LCStream class methods and properties.

Overview

The LCStream class represents text and binary datatypes. A stream value is a variable length list of characters or bytes. Streams come in two basic types, text and binary, represented by the format of the stream. Specific format information indicates either the character set (for text) or the binary format (for binary).

In addition to specific text formats, there is also the option to designate a text stream as "Native," or LCSTREAMFMT_NATIVE, indicating the characters of the stream should be stored in the local platform specific character set. (Note that the LSX LC uses unicode for text representation. To create a unicode stream, use LCSTREAMFMT_UNICODE.) Likewise, in addition to the basic binary designation (BLOB or LCSTREAMFMT_BLOB), there are four specialized binary formats:

- LCSTREAMFMT_COMPOSITE - Notes composite (Notes Rich Text format)
- LCSTREAMFMT_TEXT_LIST - list of LMBCS text strings
- LCSTREAMFMT_NUMBER_LIST - list of double precision floating point values and ranges
- LCSTREAMFMT_DATETIME_LIST - list of LCDatetime values and ranges

There are special methods dedicated to working with the three "LIST" formats. (The maximum storage size of the "LIST" format stream object is 64K.)

Type TEXT format and values

Type constant	LCTYPE_TEXT
Description	Locale-specific character stream
Other	Maximum Length = 4 Gb
	Formats: LMBCS, Native, or any valid character set (see Appendix D)

Type BINARY format and values

Type constant	LCTYPE_BINARY
Value	LCSTREAM structure
Description	Optionally formatted byte stream
Other	Maximum Length = 4 Gb
	Formats: BLOB (unformatted), Notes rich text, Notes text list, Notes number list, Notes datetime list

A stream value contains the following information:

Maximum Length	The maximum valid data length for this stream. Any value is valid, with a value of zero indicating no maximum length.
Stream Flags	Zero or more stream flags OR-ed together. See Stream Flags description below.
Stream Format	Stream data format. See Stream Format description below. If stream format is not a list type, it is a one element string or a binary stream.
Data Length	Length, in bytes, of data in the data buffer field.

Stream flags

The StreamFlags of a stream determine the behavior of the stream under particular circumstances:

LCSTREAMF_FIXED	Buffer length is always MaxLength, and the buffer is allocated once and never changed. MaxLength cannot be zero. Without this flag, the buffer is dynamically reallocated to accommodate assigned values (within MaxLength).
LCSTREAMF_TRUNCATE	Stream will automatically truncate assigned and converted values if the stream cannot accommodate them. Without this flag, assigning a value too long for the stream will generate an overflow error.
LCSTREAMF_NO_CASE	Stream is not case-sensitive. During text stream comparisons, a non case-sensitive comparison is done when either or both streams have this flag set. Without this flag, text to text comparisons are case sensitive.
LCSTREAMF_NO_TRIM	Stream should not be trimmed. This will notify Connectors and the Trim method that trimming of trailing spaces should not be performed for this stream.

Stream format

The Format of a stream indicates the structure of the stream data. The flag LCSTREAMFMTF_BINARY indicates whether the stream is of a binary format or a text format. Text formats are represented by either a character set constant or the following special value:

LCSTREAMFMT_NATIVE	The same as the native character set of the local machine
--------------------	---

For a complete list of supported character sets, see Appendix D.

Binary formats must be one of the following values:

LCSTREAMFMT_BLOB	Unformatted (<u>B</u> inary <u>L</u> arge <u>O</u> bject)
LCSTREAMFMT_COMPOSITE	Lotus Notes format Composite (also known as Rich Text or Compound Text)
LCSTREAMFMT_TEXT_LIST	Lotus Notes format Text List (multi-value list of text values)

Overview

LCSTREAMFMT_NUMBER_LIST	Lotus Notes format Number List (multi-value list of number values and ranges)
LCSTREAMFMT_DATETIME_LIST	Lotus Notes format Datetime List (multi-value list of datetime values and ranges)

Conversion is supported between stream formats excluding certain conversions. If composite is involved in a conversion, it must be the source and the target must be BLOB or a text format. In addition, conversion between number list and datetime list is not supported.

Stream Buffer

The maximum length of a stream indicates the maximum valid length in bytes for a value assigned to this stream. This can be any value up to 4 Gb. A value of zero indicates that the stream has no maximum length.

LCStream Properties

Flags	Long. The flags for the stream.
Format	Long. The stream data format.
Length	Long. The length of the stream data.
MaxLength	Long. The maximum valid data length for the stream. Any value is valid, with a value of zero indicating no maximum length. [Read-Only]
Text	String. Text representation of the stream.
Value	Variant. An array. If contents of Stream is a string , it's a one element array. If numbers or datatimes, it is an array of numbers. It can also be an array of strings for TextLists.
ValueCount	Long. Number of elements in a stream. Valid only for List<Type> formats. [Read-Only]
RangeCount	Long. Range of elements in a stream. Valid only for List<Type> formats. [Read-Only]

New method for LCStream

This is the constructor method for the LCStream class. It creates an empty LCStream object and optionally assigns initial properties.

Defined In

LCStream

Syntax

Dim *variableName* As **New** LCStream(*maxLength*, *flags*, *format*)

Parameters

<i>maxLength</i>	Long. Optional. Maximum length that this stream's data can be. A value of zero indicates no maximum, which is not valid with the flag LCSTREAMF_FIXED. Default is 0.
<i>streamFlags</i>	Long. Optional. Flags for this stream. When using the flag LCSTREAMF_FIXED to create a fixed-length stream, the stream's data buffer is allocated to be maxLength bytes. Refer to the Stream Flags section for a description of the flags. The default is 0.
<i>format</i>	Long. Optional. Initial stream format to be assigned to the stream. The default is LCSTREAMFMT_UNICODE.

Clear method for LCStream

This method clears a stream value and properties.

Defined In

LCStream

Syntax

lcStream.Clear

Example

```
Option Public
Uselsx "**lsxlc"

Sub Initialize
  Dim message As New LCStream

  message.Text = "Hello World"
  Message.Clear
  If (Len(message.Text) > 0) Then
    Print "the message is " & message.Text
  Else
    Print "the message is blank"
  End If
End Sub
```

Example Output

the message is blank

Append method for LCStream

This method appends one stream to another to yield a third LCStream object containing the data from both.

Defined In

LCStream

Syntax

Call *newStream*.**Append**(*stream1*, *stream2*)

Parameters

<i>stream1</i>	LCStream. The stream to which you want to append. The <i>newStream</i> will have this stream's format.
<i>stream2</i>	LCStream. The stream to append to <i>stream1</i> . If this stream is a different format than <i>stream1</i> , it will first be converted to the format of <i>stream1</i> .

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim stream1 As New LCStream (64, , LCSTREAMFMT_ASCII)
```

```
    Dim stream2 As New LCStream (64, , LCSTREAMFMT_ASCII)
```

```
    Dim together As New LCStream (64, , LCSTREAMFMT_ASCII)
```

```
    stream1.Text = "The quick brown fox "
```

```
    stream2.Text = "jumped over the lazy dog."
```

```
    Call together.Append (stream1, stream2)
```

```
    Print "The combined stream is " & together.Text
```

```
End Sub
```

Example Output

The combined stream is The quick brown fox jumped over the lazy dog.

Compare method for LCStream

This method compares two LCStream objects to determine if they are equal or unequal. If the streams are of different formats, they will first be converted to the same format. If both streams are text, they may be converted to unicode for comparison. In a text comparison, if either stream has the flag LCSTREAMF_NO_CASE set, then a case-insensitive text comparison will be done.

Defined In

LCStream

Syntax

result = *thisStream*.**Compare**(*baseStream*)

Parameters

baseStream LCStream. The stream to which you want to compare *thisStream*.

Return Value

result Result of the comparison:

Result > 0 (positive): *thisStream* is greater than *baseStream*.

Result < 0 (negative): *thisStream* is less than *baseStream*.

Result = 0: *thisStream* is equal to *baseStream*.

Example

```
Option Public
Option Explicit
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim stream1 As New LCStream (64, LCSTREAMF_NO_CASE, LCSTREAMFMT_ASCII)
```

```
    Dim stream2 As New LCStream (64, , LCSTREAMFMT_ASCII)
```

```
    Dim match As Long
```

```
    stream1.Text = "The quick brown fox"
```

```
    stream2.Text = "the QuiCK BROWn fOX"
```

```
    match = stream1.Compare (stream2)
```

```
    If (match = 0) Then
```

```
        Print "The first string is the same as the second."
```

```
    ElseIf (match > 0) Then
```

```
        Print "The first string comes before the second."
```

```
    Else
```

```
        Print "The first string comes after the second."
```

```
    End If
```

```
End Sub
```

Example Output

The first string is the same as the second.

Convert method for LCStream

This method obtains a stream in a particular stream format.

Defined In

LCStream

Syntax

Call *newStream.Convert* (*srcStream*, *flags*, *format*)

Parameters

srcStream

LCStream. Stream whose data is to be converted.

flags

Long. Flags that determine aspects of the conversion. Zero or more of the following values, OR-ed together:

LCCONVERTF_REFERENCE

Allows the destination stream to reference the source stream if they are of the same format. Otherwise, a data copy is always required. See Usage Notes below.

LCCONVERTF_PRESERVE

Preserves the destination stream meta-information (MaxLength and StreamFlags). Otherwise, they are taken from the source stream.

LCCONVERTF_FORCE_TEXT

Overrides the DECSTranslation setting of 0 or 1 in the NOTES.INI file for text translation and forces it to occur between any different text character sets.

format

Long. Stream format of the destination data.

Usage Notes

The use of the flag `LCCONVERTF_REFERENCE` supports an efficient method of obtaining stream data in a particular format. When requesting a stream in a specific format and using this flag, if the stream data is already in the same format, no data copying is done. The new stream simply references the data of the original stream.

When working with a stream that references another stream's data, do delete the original stream until the referencing stream is no longer needed.

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim msga As New LCStream (0, 0, LCSTREAMFMT_ASCII)
```

```
    Dim msgu As New LCStream
```

```
    msga.Text = "Peter Pan lived in Never Never Land."
```

```
    Call msgu.Convert (msga, 0, LCSTREAMFMT_UNICODE)
```

```
    Print "The length of the msg in ASCII is " & msga.Length & " while the length in Unicode is " & msgu.Length
```

```
End Sub
```

Example Output

The length of the msg in ASCII is 36 while the length in Unicode is 72

Copy method for LCStream

This method copies the value one LCStream object to another.

Defined In

LCStream

Syntax

Set newStream = lcStream.Copy

Parameters

<i>lcStream</i>	LCStream. Source stream to be copied.
-----------------	---------------------------------------

Return Value

<i>newStream</i>	LCStream. The copy of the original stream.
------------------	--

Example

```
Option Public
Uselsx "*lsxc"

Sub Initialize
    Dim msg As New LCStream
    Dim newmsg As LCstream

    msg.Text = "Peter Pan"
    Set newmsg = msg.Copy
    Print "The original msg is " & msg.Text
    Print "The new msg is " & newmsg.Text
End Sub
```

Example Output

```
The original msg is Peter Pan
The new msg is Peter Pan
```

Extract method for LCStream

This method creates a stream from part of the data of an existing stream.

Since this method works off of byte counts, it is most useful if you know you have single or double byte character sets. It's not as useful if there's a mix, like LMBCS or CodePage932 (mix of single and double byte character sets).

Defined In

LCStream

Syntax

Call *lcStream.Extract* (*srcStream*, *offset*, *length*)

Parameters

<i>srcStream</i>	LCStream. Stream supplying the data from which the new stream is created.
<i>offset</i>	Long. Byte position in <i>SrcStream</i> of the start of the new stream. If the offset exceeds the length of the stream data, the new stream is cleared.
<i>length</i>	Long. Length in bytes of the new stream copied from <i>SrcStream</i> . If the length copies more bytes than are available, the copy stops at the end of the source data.

Example

```
Option Public  
Uselx "**lsxc"
```

```
Sub Initialize
```

```
    Dim msg As New LCStream ( , , LCSTREAMFMT_ASCII)
```

```
    Dim part As New LCStream
```

```
    msg.Text = "the quick brown fox jumped over the lazy dog"
```

```
    Call part.Extract (msg, 11, 5)
```

```
    Print "The 5 bytes, starting at the 11th byte is " & part.Text
```

```
End Sub
```

Example Output

The 5 bytes, starting at the 11th byte is brown

Merge method for LCStream

This method combines one stream into another, producing a new stream.

Defined In

LCStream

Syntax

Call `lcStream.Merge(stream1, offset, stream2)`

Parameters

<i>stream1</i>	LCStream. <i>stream2</i> is combined into <i>stream1</i> .
<i>offset</i>	Long. Position in <i>lcStream</i> of the first byte of <i>stream2</i> : <i>stream2</i> is inserted into <i>stream1</i> starting at the position indicated by <i>offset</i> .
<i>stream2</i>	LCStream. <i>stream2</i> is combined into <i>stream1</i> . If <i>stream2</i> is a different format than <i>stream1</i> , it is converted to the format of <i>stream1</i> before being merged.

Example

```
Option Public
```

```
Uselsx "**lsxc"
```

```
Sub Initialize
```

```
    Dim msg As New LCStream      ' Unicode is the default format for streams
```

```
    Dim part As New LCStream
```

```
    Dim newmsg As New LCStream
```

```
    msg.Text = "the quick brown fox jumped over the lazy dog"
```

```
    part.Text = "very "
```

```
    REM counting start with 1 and each character in a unicode string is 2 bytes
```

```
    Call newmsg.Merge (msg, 9, part)
```

```
    Print "The message, after inserting '" & part.Text"' starting at the 9th byte, is '" & newmsg.Text
End Sub
```

Example Output

The message, after inserting 'very ' starting at the 9th byte, is the very quick brown fox jumped over the lazy dog

ResetFormat method for LCStream

This method resets the format an LCStream object, without affecting the data or other properties.

This method may be useful when storing multiple language strings in a single database and you need to switch the format. A typical case would be web browsers accessing an application from different countries and requiring different translations of the content. The different translations could be stored in a single database. Since most RDBMS environments do not permit multiple languages within a single database, the translations could be selected based on a key. The contents would be treated as blobs and the format 'reset' to the correct character set before being passed to the web browser.

Defined In

LCStream

Syntax

Call *lcStream*.**ResetFormat** (*format*)

Parameters

format

Long. The new format for the stream object. See the list of stream formats in the section “Stream Formats” earlier in this chapter.

Example

Option Public

Uselsx "**lsxlc"

Sub Initialize

Dim msg As New LCStream (0, 0, LCSTREAMFMT_ASCII)

msg.Text = "Peter Pan lived in Never Never Land."

Print "The length of the msg is " & msg.Length

Call msg.ResetFormat (LCSTREAMFMT_UNICODE)

Print "The length of the msg and its contents have not changed but the format is now Unicode"

Print "The length of the msg is " & msg.Length

End Sub

Example Output

The length of the msg is 36

The length of the msg and its contents have not changed but the format is now Unicode

The length of the msg is 36

SetFormat method for LCStream

This method sets the format for an LCStream object, converting the stream data if necessary. This method may be more efficient than the Convert method, since a second stream is not required.

Defined In

LCStream

Syntax

Call *lcStream.setFormat* (*format*)

Parameters

<i>format</i>	Long. The format to which to set the stream object. See the list of stream formats in the section “Stream Formats” earlier in this chapter. The data will be converted to this stream format.
---------------	---

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize  
    Dim msg As New LCStream (0, 0, LCSTREAMFMT_ASCII)  
  
    msg.Text = "Peter Pan lived in Never Never Land."  
    Print "The length of the msg in ASCII is " & msg.Length  
    Call msg.SetFormat (LCSTREAMFMT_UNICODE)  
    Print "The length of the msg in Unicode is " & msg.Length  
End Sub
```

Example Output

```
The length of the msg in ASCII is 36  
The length of the msg in Unicode is 72
```

Trim method for LCStream

This method trims trailing spaces from text streams. If the stream is not a text format or if the stream has the flag `LCSTREAM_NO_TRIM` set, then this method will perform no action.

Defined In

LCStream

Syntax

lcStream.Trim

Example

```
Option Public  
Uselsx "**!sxlc"
```

```
Sub Initialize  
  Dim msg As New LCStream
```

```
  msg.Text = "  this has space ar the start and the end  "  
  msg.Trim ' trim trailing spaces (not leading ones)  
  Print "The msg is *" & msg.Text & "*"  
End Sub
```

Example Output

The msg is * this has space ar the start and the end*

DatetimeListGetRange method for LCStream

This method gets a range of values in an LCStream DatetimeList object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimeListGetRange* (*index*, *startDatetime*, *endDatetime*)

Parameters

<i>index</i>	Long. Index position of the range in the datetimeList stream.
<i>startDatetime</i>	LCDatetime. Output. The starting datetime of the range.
<i>endDatetime</i>	LCDatetime. Output. The ending datetime of the range.

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize
```

```
    Dim datelist As New LCStream (0, 0, LCSTREAMFMT_DATETIME_LIST)
```

```
    Dim start As New LCDatetime
```

```
    Dim finish As New LCDatetime
```

```
    datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000 12:00AM," & _  
    "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
    Call datelist.DatetimeListGetRange (1, start, finish)
```

```
    Print "The new 1st range is " & start.Text & " - " & finish.Text
```

```
End Sub
```

Example Output

The new 1st range is 05:00:00 PM - 06:00:00 PM

DatetimeListGetValue method for LCStream

This method retrieves a datetime value from a specified place in a DateTimeList LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimeListGetValue(index, datetime)*

Parameters

<i>index</i>	Long. Index position of the datetime value to retrieve.
<i>datetime</i>	LCDateTime. The datetime value.

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize
```

```
    Dim datelist As New LCStream (0, 0, LCSTREAMFMT_DATETIME_LIST)
```

```
    Dim num As New LCDatetime
```

```
    datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000 12:00AM," & _  
"5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
    Call datelist.DatetimeListGetValue (3, num)
```

```
    Print "The new 3rd datetime is " & num.Text
```

```
End Sub
```

Example Output

The new 3rd datetime is 12/31/1999 12:59:00 PM

DatetimeListInsertRange method for LCStream

This method inserts a datetime range into a datetime list stream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *datetimeNew.DatetimeListInsertRange* (*index*, *startDatetime*, *endDatetime*)

Parameters

<i>index</i>	Long. The position at which to insert the value(s).
<i>startDatetime</i>	LCDatetime. The first datetime value for the range.
<i>endDatetime</i>	LCDatetime. The second datetime value for the range.

Example

Option Public

Uselsx "**Isxlc"

Sub Initialize

Dim datelist As New LCStream (0, 0, LCSTREAMFMT_DATETIME_LIST)

Dim start As New LCDateTime

Dim finish As New LCDateTime

start.Text = "8/6/1941"

finish.Text = "7/1/1997"

datelist.Text = "12/25/50, 7:00AM, 12/31/99 12:59PM, 1/1/2000 12:00AM, 5:00PM - 6:00PM,
6:30AM, 5/1/96 - 5/31/96"

Call datelist.DatetimeListInsertRange (2, start, finish)

Print "The new stream is " & datelist.Text

End Sub

Example Output

The new stream is 12/25/1950, 07:00:00 AM, 12/31/1999 12:59:00 PM, 01/01/2000 12:00:00 AM,
06:30:00 AM, 05:00:00 PM - 06:00:00 PM, 08/06/1941 - 07/01/1997, 05/01/1996 - 05/31/1996

DatetimeListInsertValue method for LCStream

This method inserts a value into a datetime list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimeListInsertValue* (*index*, *datetime*)

Parameters

<i>index</i>	Long. The index position at which to insert the value.
<i>datetime</i>	LCDatetime. The datetime value to insert.

Example

```
Option Public  
Uselsx "*Isxlc"
```

```
Sub Initialize
```

```
    Dim datelist As New LCStream (0, 0, LCSTREAMFMT_DATETIME_LIST)
```

```
    Dim clock As New LCDatetime
```

```
    clock.SetCurrent
```

```
    datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000 12:00AM," & _  
    "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
    Call datelist.DatetimeListInsertValue (3, clock)
```

```
    Print "The new stream is " & datelist.Text
```

```
End Sub
```

Example Output

The new stream is 12/25/1950, 07:00:00 AM, 09/08/1998 05:22:23.96 PM, 12/31/1999 12:59:00 PM, 01/01/2000 12:00:00 AM, 06:30:00 AM, 05:00:00 PM - 06:00:00 PM, 05/01/1996 - 05/31/1996

DatetimestreamRemoveRange method for LCStream

This method removes a range from a datetime list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimestreamRemoveRange* (*index*)

Parameters

index Long. The position of the value to remove.

Example

```
Option Public  
Uselsx "*Isxlc"
```

```
Sub Initialize
```

```
    Dim datelist As New LCStream (0, 0, LCSTREAMFMT_DATETIME_LIST)
```

```
    datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000 12:00AM," & _  
    "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
    Call datelist.DatetimeListRemoveRange (2)
```

```
    Print "The new stream is " & datelist.Text
```

```
End Sub
```

Example Output

The new stream is 12/25/1950, 07:00:00 AM, 12/31/1999 12:59:00 PM, 01/01/2000 12:00:00 AM,
06:30:00 AM, 05:00:00 PM - 06:00:00 PM

DatetimeListRemoveValue method for LCStream

This method removes a value from a datetime list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimeListRemoveValue(index)*

Parameters

index Long. The position of the value to remove.

Example

```
Option Public
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim datelist As New LCStream (0, 0, LCSTREAMFMT_DATETIME_LIST)
```

```
    datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000 12:00AM," & _
    "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
    Call datelist.DatetimeListRemoveValue (3)
```

```
    Print "The new stream is " & datelist.Text
```

```
End Sub
```

Example Output

```
The new stream is 12/25/1950, 07:00:00 AM, 01/01/2000 12:00:00 AM, 06:30:00 AM, 05:00:00
PM - 06:00:00 PM, 05/01/1996 - 05/31/1996
```

NumberListGetRange method for LCStream

This method selects a particular range from a number list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream.NumberListGetRange* (*index*, *startNumber*, *endNumber*)

Parameters

<i>index</i>	Long. The position of the range to select.
<i>startNumber</i>	Double. Output. The first number value for the range.
<i>endNumber</i>	Double. Output. The second number value for the range.

Example

```
Option Public  
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)
```

```
    Dim start As Double
```

```
    Dim finish As Double
```

```
    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"
```

```
    Call numlist.NumberListGetRange (1, start, finish)
```

```
    Print "The new 1st range is " & start & " - " & finish
```

```
End Sub
```

Example Output

The new 1st range is 50 - 55

NumberListGetValue method for LCStream

This method retrieves a specified value from a number list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream.NumberListGetValue* (*index*, *number*)

Parameters

<i>index</i>	Long. The index position of the value to retrieve.
<i>number</i>	Double. The variable in which to place the value.

Example

```
Option Public
Useslx "**lsxlc"

Sub Initialize
    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)
    Dim num As Double

    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"
    Call numlist.NumberListGetValue (3, num)
    Print "The new 3rd number is " & num
End Sub
```

Example Output

The new 3rd number is 33

NumberListInsertRange method for LCStream

This method inserts a number range into a number list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream.NumberListInsertRange(index, startNumber, endNumber)*

Parameters

<i>index</i>	Long. The index position at which to insert the value.
<i>startNumber</i>	Double. The first number of the range.
<i>endNumber</i>	Double. The second number of the range.

Example

```
Option Public  
Uselsx "*lsxlc"
```

```
Sub Initialize  
    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)  
  
    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"  
    Call numlist.NumberListInsertRange (2, 8.8, 9.9)  
    Print "The new stream is " & numlist.Text  
End Sub
```

Example Output

The new stream is 1.11, 22.2, 33, 4.444, 66, 50 - 55, 8.8 - 9.9, 77 - 79

NumberListInsertValue method for LCStream

This method inserts a value into a number list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream.NumberListInsertValue(index, number)*

Parameters

index Long. The position at which to insert the value.

number Double. The value to insert.

Example

```
Option Public
Usesx "*lsxlc"
```

```
Sub Initialize
  Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

  numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"
  Call numlist.NumberListInsertValue (3, 99.99)
  Print "The new stream is " & numlist.Text
End Sub
```

Example Output

The new stream is 1.11, 22.2, 99.99, 33, 4.444, 66, 50 - 55, 77 - 79

NumberListRemoveRange method for LCStream

This method removes a range of numbers as indicated by an index from a number list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream.NumberListRemoveRange(index)*

Parameters

index Long. The position of the range to remove.

Example

```
Option Public
Uselsx "*lsxlc"

Sub Initialize
    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"
    Call numlist.NumberListRemoveRange (2)
    Print "The new stream is " & numlist.Text
End Sub
```

Example Output

The new stream is 1.11, 22.2, 33, 4.444, 66, 50 - 55

NumberListRemoveValue method for LCStream

This method removes a value at a specified position from a number list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream.NumberListRemoveValue(index)*

Parameters

index Long. The position of the value to remove.

Example

```
Option Public
Uselsx "*lsxlc"

Sub Initialize
    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"
    Call numlist.NumberListRemoveValue (3)
    Print "The new stream is " & numlist.Text
End Sub
```

Example Output

The new stream is 1.11, 22.2, 4.444, 66, 50 - 55, 77 - 79

TextListFetch method for LCStream

This method fetches a text list from a text list stream object and assigns it to another stream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_TEXT_LIST.

Defined In

LCStream

Syntax

Call *lcStream.TextListFetch(index, format, stream)*

Parameters

<i>index</i>	Long. The index position of the value to fetch. If the index is one, this method will accept a text stream as a single-entry text list.
<i>format</i>	Long. The format of the stream to return.

Return Value

<i>stream</i>	LCStream. The new stream object.
---------------	----------------------------------

Example

```
Option Public
```

```
Uselsx "**lsxlc"
```

```
Sub Initialize
```

```
    Dim textlist As New LCStream (0, 0, LCSTREAMFMT_TEXT_LIST)
```

```
    Dim text As String
```

```
    textlist.text = "red, orange, yellow, green, blue, indigo, violet"
```

```
    Call textlist.TextListFetch (3, text)
```

```
    Print "The 3rd string in the text list is " & text
```

```
End Sub
```

Example Output

The 3rd string in the text list is yellow

TextListInsert method for LCStream

This method inserts text or a text list into a text list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_TEXT_LIST.

Defined In

LCStream

Syntax

Call *lcStream.TextListInsert(index, stream)*

Parameters

<i>index</i>	Long. The position at which to insert the text string.
<i>stream</i>	LCStream. The text string to insert.

Example

```
Option Public  
Uselsx "*lsxc"
```

```
Sub Initialize  
  Dim textlist As New LCStream (0, 0, LCSTREAMFMT_TEXT_LIST)  
  
  textlist.text = "red, orange, yellow, green, blue, indigo, violet"  
  Call textlist.TextListInsert (3, "black")  
  Print "The new value of the text list is " & textlist.text  
End Sub
```

Example Output

The new value of the text list is red, orange, black, yellow, green, blue, indigo, violet

TextListRemove method for LCStream

This method removes a string from a text list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_TEXT_LIST.

Defined In

LCStream

Syntax

Call *lcStream.TextListRemove(index)*

Parameters

index Long. Position of the text stream to remove.

Example

```
Option Public
Uselsx "*lsxlc"

Sub Initialize
    Dim textlist As New LCStream (0, 0, LCSTREAMFMT_TEXT_LIST)

    textlist.text = "red, orange, yellow, green, blue, indigo, violet"
    Call textlist.TextListRemove (3)
    Print "The new value of the text list is " & textlist.text
End Sub
```

Example Output

The new value of the text list is red, orange, green, blue, indigo, violet

Appendix A

Error Messages

This appendix provides information about error messages.

Each entry each provides the error code in both hexadecimal and decimal format (hexadecimal is first, decimal second), the LCSTATUS constant (usable from any language), the error text for the message, and then a description of the error.

Parameterized elements of a message are indicated in all uppercase, and are replaced at error generation time with the relevant value. Optional elements of a message are enclosed in square brackets, and are included only if the contained parameters have values provided.

Anatomy of an Error Message

This section provides information about how to interpret the LSX LC error messages. It describes the format of each error and what each component of the error message indicates.

Example Error Message

Error: Two Fields cannot have the same name within a Form, Connector 'notes', Method -Create [Metadata]- (0x80000803)

Error Message Components

"Error: " - This prefix is added to any error. It is not included for 'events' (informational messages).

"Two Fields cannot have the same name within a Form" - This is the specific error message. This can be either an LC error message, or an external error message. In this case, it is an external error message.

"Connector 'Notes'" - This is the Connector which generated the error message. This element is only included when the error occurs inside an LCConnection method - it is skipped otherwise.

General Errors

“Method -Create [Metadata]” - This is the LCConnection method in which the error occurred. It is only generated when the Connector is included in the error message. It indicates the LCConnection method generating the error. In addition, for methods which accept a parameter indicating an object type (Create, Drop, or Catalog) or action type (Action), the object or action type is included in square brackets for more information.

“(0x80000803)” - This is the external error code generated by the backend system. It is only included when the actual error was LCFAIL_EXTERNAL - which is the case whenever the error is not an LC error. When the value is between -65536 and 65536, the decimal value is used. When it is lower or higher, the hex value is used.

General Errors

Format of General Errors

Errors are in the following format:

Hex Value	Decimal Value	Constant	Text
Description			

General Errors

&H101 257	LCFAIL_PROGRAM	Lotus Connector program failure - contact Lotus support
-----------	----------------	---

An unexpected internal error has occurred. Collect all information regarding events leading up to this error and contact Lotus technical support.

&H102 258	LCFAIL_MEMORY	Unable to allocate memory
-----------	---------------	---------------------------

An attempt to allocate program memory failed. There are a few possible causes of this error. First check system resources to determine if your system is low on memory. If it is, then more memory is needed on your system to support the processing tasks occurring. If your system still has significant memory available, then a request for a very large amount of memory was made. Certain operations are memory intensive, such as loading extremely large data values, requesting too many records at one time, or using a metaconnector such as the order metaconnector (which loads the entire result set at once) against a very large result set.

Connector Errors

&H3001 12289 LCFAIL_UNAVAILABLE Requested functionality is not available

A request for unsupported functionality was made. This can occur when using a Connector which doesn't support a specific operation (for example, a Connector may not support the Create method), or when an LC API method doesn't support a specific request (see the documentation for the method in question).

&H3002 12290 LCFAIL_END_OF_DATA The last data value has been retrieved

The last element has been retrieved from a list of available data values. This is a normal return code from any operation which is called to iterate through data values (for example, `LCConnection.Fetch`, `LCConnect.ListProperty`, `LCFieldlist.List`, etc...). It does not indicate a fatal error, and is usually part of a normal flow of operations.

&H3003 12291 LCFAIL_INVALID_INDEX Cannot locate list element

A request made of a multi-value object provided an invalid index. This normally occurs when accessing a list entity (for example, a field in a fieldlist or a text entry in a text list) where the index provided is greater than the maximum valid index for that object. Note that all indices are one-based - so that an object with three elements has valid index values of 1, 2, and 3.

&H3004 12292 LCFAIL_INVALID_LIST Invalid List direction

When using a method which iterates through values, an `LCLIST` parameter indicates the listing method (for example, start at the first, or return the next entry). Either an invalid listing direction was provided, or a valid direction was provided, but is not supported in the current context.

&H3005 12293 LCFAIL_INVALID_CONVERT Invalid conversion

An unsupported data conversion was attempted. Conversion is valid between members of the same data type "class" (numbers, streams, and datetimes) as well as to and from text. Conversions between datatype classes where one type is not text (for example, converting between a datetime and a currency) will return this error. In addition, conversion to or from a binary stream formats is generally only valid when one type is text or a binary BLOB format.

&H3006 12294 LCFAIL_INVALID_TEXT_LIST This operation requires a valid text list

An action that operates on a text list stream (a binary stream of format `LCSTREAMFMT_TEXT_LIST`) was provided a stream of a different format or with an invalid structure. Ensure that the stream data is a properly formatted text list.

&H3007 12295 LCFAIL_INVALID_NUMBER_LIST This operation requires a valid number list

General Errors

An action that operates on a number list stream (a binary stream of format LCSTREAMFMT_NUMBER_LIST) was provided a stream of a different format or with an invalid structure. Ensure that the stream data is a properly formatted number list.

&H3008 12296 LCFAIL_INVALID_DATETIME_LIST This operation requires a valid datetime list

An action that operates on a datetime list stream (a binary stream of format LCSTREAMFMT_DATETIME_LIST) was provided a stream of a different format or with an invalid structure. Ensure that the stream data is a properly formatted datetime list.

&H3009 12297 LCFAIL_ZERO_INDEX All index values are one-based - an index of zero is not valid

A request made of a multi-value object provided an index of zero. All objects and methods which support an index reference (for example, fieldlists and text lists) use one-based indices - zero is never a valid index value.

&H300A 12298 LCFAIL_ZERO_COUNT This operation requires a non-zero count

A parameter indicating a count was given a value of zero when it is not valid. Zero counts are only valid in specific cases - check the method documentation for more information.

&H300B 12299 LCFAIL_ZERO_OFFSET All offset values are one-based - an offset of zero is not valid

A request made of a multi-value object provided an offset of zero. All methods which support an offset parameter (for example, LCStream.Merge) use one-based offsets - zero is never a valid offset value.

&H300C 12300 LCFAIL_ZERO_FORMAT This operation requires a non-zero Stream Format

A parameter indicating a stream format was given a value of zero when it is not valid. Zero stream formats are only valid in specific cases - check the method documentation for more information. A list of valid stream formats is provided in your documentation.

&H300D 12301 LCFAIL_NULL_BUFFER A NULL buffer was provided when one was required

A method with a required buffer parameter was provided no value where a valid buffer was required. Check the method documentation for more information.

&H300E 12302 LCFAIL_NULL_RESULT A return parameter is required, but none was provided

A method with a required output parameter was provided no value where a valid parameter was required. Check the method documentation for more information.

&H300F 12303 LCFAIL_FIXED_LENGTH A fixed-length stream requires a non-zero length

When requesting a fixed-length stream (with the stream flag LCSTREAMF_FIXED), a maximum length property with a non-zero value must be provided via LCStream.Create

&H3010 12304 LCFAIL_INVALID_FLAGS The supplied flags are invalid, possibly due to a conflict

Bitwise flags provided for an operation are not valid - either invalid flags were specified, or conflicting flags were provided. Check the object or method documentation for more information on valid flags.

&H3011 12305 LCFAIL_TEXT_TRANSLATE Text translation failure

Translation between character sets failed. This may be caused by an incorrect character set indicator or invalid data in the source text. If the problem persists, contact Lotus technical support.

&H3012 12306 LCFAIL_NULL_FIELDNAME A NULL field name was provided

A NULL or empty field name was provided when a valid field name was required. Ensure that the field name being provided has at least one character, and check the method documentation for more information.

&H3013 12307 LCFAIL_INVALID_FIELDLIST Invalid fieldlist

An LCFIELDLIST object handle is either not a valid handle, or is zero. Object handles must be a valid value returned by a method which creates a new fieldlist. Handles are no longer valid once the object is freed.

&H3014 12308 LCFAIL_INVALID_CONNECTION Invalid connection

An LCCONNECTION object handle is either not a valid handle, or is zero. Object handles must be a valid value returned by a method which creates a new connection. Handles are no longer valid once the object is freed.

&H3015 12309 LCFAIL_EMPTY_FIELDLIST This operation cannot be performed on a fieldlist with no fields

A fieldlist that contains no fields is not valid in this context. To avoid this error, ensure that any fields are added to the fieldlist, and check the method documentation for more information.

General Errors

&H3016 12310 LCFAIL_NAME_FIELDLIST This operation cannot be performed on a name-only fieldlist

A fieldlist created with a record count of zero is not valid in this context. Such fieldlists are intended for name mapping, and cannot contain data values. As such, they cannot be used in any situation where data will be read from or written to a fieldlist. To avoid this error, provide a non-zero record count when creating the fieldlist.

&H3017 12311 LCFAIL_NATIVE_OVERRIDE Native text format override supplied is not a valid stream format indicator

An override of the native character set was provided in an .INI variable as indicated in the documentation. The character set indicated is not a valid character set identifier. Check the documentation for a list of valid character set strings.

&H3018 12312 LCFAIL_RECORD_INDEX An invalid fieldlist record index was encountered

A fieldlist record index is not valid for the relevant fieldlist. This generally occurs with LCConnection class methods which accept a fieldlist and a record index. The index must be at least one (all indices are one-based), and cannot be greater than the number of records that the fieldlist was created with.

&H3019 12313 LCFAIL_RECORD_COUNT Request to transfer more records than allocated in fieldlist

A fieldlist record count is not valid for the relevant fieldlist. This generally occurs with LCConnection class methods which accept a fieldlist and a record count. The count plus the record index cannot exceed the number of records that the fieldlist was created with. In addition, a count of zero is only valid in specific situations - see the method documentation for more information.

&H301A 12314 LCFAIL_LIST_SETUP Fieldlist iteration requires initial setup

Listing fieldlist in a fieldlist with LCFieldlist.List requires that LCFieldlist.ListSetup be called first to prepare the iteration.

&H301B 12315 LCFAIL_NO_MERGE_DATA The data fieldlist in a merge cannot be a name-only fieldlist

When using the methods LCFieldlist.Merge or MergeVirtual, the fieldlist provided as the data fieldlist cannot be a name-only fieldlist. It must be created with a non-zero record count.

&H301C 12316 LCFAIL_NO_RESULTSET This operation requires an active result set

A requested LCConnection operation cannot be performed unless the connection has an active result set. Produce a result set before attempting this operation. This error is generally produced by calling LCConnection.Fetch without a previous call to a method which produces a result set (such as Execute, Select, Call, or Catalog).

&H301D 12317 LCFAIL_NO_WRITEBACK This operation requires an active writeback result set

A requested LCConnection operation cannot be performed unless the connection has an active writeback result set. Produce a writeback-enabled result set before attempting this operation. This error is generally produced by calling LCConnection.Update or Remove with the “Writeback” property set, but without a previous call to a method which produces a writeback result set (such as Execute or Select with the “Writeback” property set). If the result set is produced without the writeback property set, then writeback operations are not supported against that result set.

&H301E 12318 LCFAIL_WRITEBACK_COUNT Writeback operation record counts must be one

When requesting a writeback operation against an LCConnection, only the most recently fetched record is affected, so the record count must be one.

&H301F 12319 LCFAIL_TRANSLATE_INIT Text translation subsystem initialization failure

The internal character set translation subsystem failed to initialize properly. This is generally due to an improperly installed environment. Check the document for installation and platform-specific notes, and contact Lotus technical support if the problem persists.

&H3020 12320 LCFAIL_TIMEBOMB This time-limited version has expired [as of DATE]

This is a time-limited version which expired on the date indicated. A new version with a later timebomb, or a complete version must be obtained from Lotus.

&H3021 12321 LCFAIL_SESSION_NOT_INIT The Lotus Connector Session must be initialized before performing any operation

The Lotus Connector Session class must be initialized before any other LC objects can be created or methods used.

&H3022 12322 LCFAIL_CONNECTOR_VERSION Incorrect Connector version

A Connector was built with an incompatible version of the Lotus Connector Toolkit. Contact Lotus or the developer of the Connector for an updated version.

General Errors

&H3023 12323 LCFAIL_NOT_CONNECTED This operation requires a connection to a Connector

The LCConnection method called cannot be used on an unconnected LCConnection. Call the Connect method before this operation. In general, any operation which produces a result set or manipulates data or metadata cannot be called before connecting - only getting and setting properties may occur. One exception for some Connectors is that a server or database catalog result set may be produced and fetched from before connecting.

&H3024 12324 LCFAIL_CONNECTED This operation cannot be performed when there is a valid connection to a Connector

The LCConnection operation performed is not valid when the LCConnection is actively connected - the Disconnect method must first be called. This may occur when performing an action which affects the connection itself, such as using the Drop method on a database.

&H3025 12325 LCFAIL_EXTERNAL <external error text>

An external database or application accessed by a Connector generated an error. The external error text and code are available with the method LCSession.GetStatus.

&H3026 12326 LCFAIL_ACTIVE_SUBCONNECTOR The sub-Connector of a metaconnector can only be set once

Any subconnector property of a metaconnector can only be set once - any attempt to reset it will generate this error.

&H3027 12327 LCFAIL_TRANSLATE_TABLE No translation tables are available for the character set

A translation table is available for most translations between supported character sets. This error indicates that the translation table for a specific character set or translation is not available and the character set translation cannot be performed. Often, a similar character set is available and can be used to obtain the same effective translation.

&H3028 12328 LCFAIL_NO_SCROLL This operation requires an active scrollable result set

Using the LCConnection.Fetch method with a negative record count is only valid when the Connector supports scrolling result sets and the active result set was produced with the "Scrolling" property set. This error is returned from the Fetch method when a negative record count is provided and either of these conditions is not met.

&H3029 12329 LCFAIL_BINARY_FORMAT This operation requires a non-binary Stream Format

Certain LCStream methods are not valid on formatted binary stream formats. This method requires a stream with a text format, or unformatted BLOB binary format. Check the method documentation for more information.

&H302A	12330	LCFAIL_ASYNC_ACTIVE	Asynchronous operation is still active
--------	-------	---------------------	--

This error is returned from methods that wait for an asynchronous operation to complete. When the asynchronous operation is still active and any wait period specified passes, this error is returned.

Parameterized Connector Errors

&H3101 12545 LCFAIL_INVALID_METADATA Metadata object
['METADATA'] does not exist

The value provided for the “Metadata” property is not a valid metadata object in the external system. The invalid metadata name will generally be provided. Ensure that a valid value is being provided - common errors include misspellings, incorrect capitalization in case-sensitive systems, and being connected to the wrong database.

&H3102 12546 LCFAIL_TYPE_MISMATCH Type mismatch [for field
‘FIELDNAME’] [; Connector: TYPE1] [, External: TYPE2]

Data type mismatch. Implicit data conversion is supported only within a datatype “class” - number, stream, or datetime. When mapping is attempted between datatypes in different classes (for example, INT to TEXT; or DATETIME to NUMERIC), this error is generated. The fieldname and datatypes are generally provided in the error text. Ensure that the datatypes being mapped are members of the same datatype class.

&H3103 12547 LCFAIL_DUPLICATE Duplicate object [‘NAME’]

An object of the same name already exists - select a different name. This error can occur when creating an API object or an external system object, such as creating a new metadata object where one of the same name already exists.

&H3104 12548 LCFAIL_FIELD_COUNT_MISMATCH Field count mismatch [;
Connector: COUNT1, External: COUNT2]

The number of fields did not match between a Connector and an external system. In some cases, this problem can be resolved automatically (for example, when writing fields to a backend system that supports more fields than provided). In other cases, this error will be generated (for example, when writing 5 fields to a backend system which supports 3 fields). To avoid this error, fields can be removed from a fieldlist on a per-operation basis by using field flags.

&H3105 12549 LCFAIL_KEY_COUNT_MISMATCH Key count mismatch [;
Connector: COUNT1, External: COUNT2]

The number of keys did not match two systems. In cases where keys must be specified for two systems, the same number of keys must be provided for both systems.

&H3106 12550 LCFAIL_STAMPFIELD_TYPE Timestamp field
['FIELDNAME'] must be type Datetime[; Actual: TYPE]

A timestamp field provided for the “StampField” property of a Connector must represent an

external field of type datetime. This error is returned when it is a number or stream datatype.

&H3107 12551 LCFAIL_FIELD_TYPE Type mismatch for field ['FIELDNAME']
used in this context [; Expected: TYPE1 [, Actual: TYPE2]]

Specific functionality requested of a Connector requires that a field be of a particular datatype. This error is generated when a specific datatype is expected for a particular field, but an incompatible datatype is provided.

&H3108 12552 LCFAIL_MERGE_FIELD Field mapping failed due to a
missing field ['FIELDNAME']

When using the methods LCFieldlist.Merge or MergeVirtual, there was a mismatch between the fieldlists. Either a field in the namelist had no match in the data list, or vice-versa. If such a mismatch occurs and the merge flag LCMERGEF_NAME_LOSS or LCMERGEF_DATA_LOSS is not specified to indicate that loss of fields is allowed, then this error is generated. Ensure that the field indicated in the error has a corresponding field in the other fieldlist.

&H3109 12553 LCFAIL_MISSING_PROPERTY No value supplied for required
property ['PROPERTY']

The indicated property requires a value for the requested operation, but none was provided. Supply a valid value for the property. Check the documentation for the Connector or the method being called (for example, when using the LCConnection.Create method to create an index, the property "Index" is required, and if no value was given, this error would be generated).

&H310A 12554 LCFAIL_PROPERTY_CONFLICT Conflicting values for
properties ['PROPERTY1' and 'PROPERTY2']

The values provided for the two indicated properties conflict with one another. Check the Connector documentation for information regarding conflicting property values for this Connector.

&H310B 12555 LCFAIL_INVALID_PROPERTY Invalid property ['PROPERTY']

The attempt to get or set the indicated property failed because it is not supported by the Connector. Check the Connector documentation for more information on supported properties for this Connector.

&H310C 12556 LCFAIL_PROPERTY_VALUE Invalid property value [for
property 'PROPERTY']

The value provided for the indicated property is not a valid value. Check the Connector documentation for information on valid property values for this Connector.

General Errors

&H310D 12557 LCFAIL_INVALID_CHARSET The text format provided
[‘CHARSET’] is not a valid Lotus Connector character set indicator

An attempt to indicate a character set by it’s textual representation failed due to an invalid string.
Check the documentation for a list of supported character set strings.

&H310E 12558 LCFAIL_READ_ONLY_PROPERTY Cannot change the value
of read-only property [‘PROPERTY’]

The indicated property is read-only for this Connector and an attempt to set it’s value was made.
For example, the “TextFormat” property of some Connectors cannot be assigned, so will
generate this error on any such attempt.

&H310F 12559 LCFAIL_MISSING_CONNECTORLCX Cannot load Connector
LCX Library [‘CONNECTOR’]

The Connector library .LCX file could not be loaded. Ensure that a valid Connector was
specified and that the Connector is available on the system. A common mistake which produces
this error is to not have any client files installed to support the Connector (for example, use of
the DB2 Connector requires that a DB2 client product be installed)

&H3110 12560 LCFAIL_INVALID_CONNECTORLCX Invalid Connector LCX
Library [‘CONNECTOR’]

The Connector library .LCX file was located and loaded, but was not a valid Connector.
Contact Lotus or the Connector vendor for a valid Connector library.

Field-Specific Parameterized Connector Errors

&H3201 12801 LCFAIL_OVERFLOW Data overflow[in field 'FIELDNAME']

A data overflow was encountered when transferring data to or from a field. This error generally occurs when writing data to an external system, and the data is too large for the target field (for example, writing the text “abcdefg” to a SQL CHAR(5) field). This error can also occur, though, when reading data into an internal datatype and the data cannot be accommodated (a number or datetime is out of range, or a stream exceeds the maximum allowed length. This error can be suppressed by assigning the field flag LCFIELDF_TRUNC_DATA, or selecting the corresponding UI option to allow data loss on overflow.

&H3202 12802 LCFAIL_PRECISION_LOSS Data precision loss [in field 'FIELDNAME']

This error is generated when performing type mapping between internal and external fields when the datatypes are different, and precision may be lost on data transfer. Note that this error indicates that precision will be lost between the datatypes, irrelevant of the actual data (for example, writing a FLOAT values of 3 to an INT field will be considered a precision loss, since it checks the datatypes once at mapping time, rather than the data values for each transfer). This error can be suppressed by assigning the field flag LCFIELDF_TRUNC_PREC, or selecting the corresponding UI option to allow precision loss.

&H3203 12803 LCFAIL_INVALID_INT Invalid integer value [in field 'FIELDNAME']

The indicated field contains an invalid integer value, or an attempt to assign an invalid integer value was made. Ensure that the data source for the field was a valid value and check that the field value is not being destroyed by an incorrect operation or method call.

&H3204 12804 LCFAIL_INVALID_FLOAT Invalid float value [in field 'FIELDNAME']

The indicated field contains an invalid floating point value, or an attempt to assign an invalid floating point value was made. Ensure that the data source for the field was a valid value and check that the field value is not being destroyed by an incorrect operation or method call.

&H3205 12805 LCFAIL_INVALID_CURRENCY Invalid currency value [in field 'FIELDNAME']

The indicated field contains an invalid currency value, or an attempt to assign an invalid currency value was made. Ensure that the data source for the field was a valid datetime value within the allowable range of an LCCurrency and check that the field value is not being destroyed by an incorrect operation or method call.

General Errors

&H3206 12806 LCFAIL_INVALID_NUMERIC Invalid numeric value [in field 'FIELDNAME']

The indicated field contains an invalid numeric value, or an attempt to assign an invalid numeric value was made. Ensure that the data source for the field was a valid numeric value within the allowable range of an LCNumeric and check that the field value is not being destroyed by an incorrect operation or method call.

&H3207 12807 LCFAIL_INVALID_DATETIME Invalid datetime value [in field 'FIELDNAME']

The indicated field contains an invalid datetime value, or an attempt to assign an invalid datetime value was made. Ensure that the data source for the field was a valid datetime value within the allowable range of an LCDatetime, and check that the field value is not being destroyed by an incorrect operation or method call.

&H3208 12808 LCFAIL_INVALID_STREAM Invalid stream value [in field 'FIELDNAME']

The indicated field contains an invalid stream value, or an attempt to assign an invalid stream value was made. Ensure that the data source for the field was a valid value and check that the field value is not being destroyed by an incorrect operation or method call.

&H3209 12809 LCFAIL_INVALID_FIELD Invalid field ['FIELDNAME']

An LCFIELD object handle is either not a valid handle, or is zero. Object handles must be a valid value returned by a method which creates a new field. Handles are no longer valid once the object is freed.

&H320A 12810 LCFAIL_INVALID_TYPE Invalid data type[for field 'FIELDNAME']

The datatype for the indicated field is an unknown datatype, or a datatype parameter was provided with an invalid value. This error will generally only occur if some sort of corruption exists, but could also be indicative of a version mismatch between a specific Connector and the calling program or tool. Check to ensure that only valid datatypes of the form LCTYPE_XXX are being used, and if the problem persists, contact Lotus technical support.

&H320B 12811 LCFAIL_INVALID_KEY Invalid key field ['FIELDNAME']

The indicated field was provided as a key field, but no such searchable field exists in the result set or external system. Two common sources of this error are misspellings of the key field name or use of a non-searchable field (as defined by the external systems - check the Connector documentation for information on datatypes that cannot be used as keys) as a key. This error can also occur when the key field is left out of the result set.

&H320C 12812 LCFAIL_DUPLICATE_KEY Duplicate key field
[‘FIELDNAME’]

The indicated field was provided as a key field twice - a single field can only be used once as a key. Either remove the second occurrence of the key field, or provide a different field name in it’s place.

&H320D 12813 LCFAIL_INVALID_STAMPFIELD Invalid timestamp field
[‘FIELDNAME’]

The indicated field was provided as a timestamp field, but no such field exists in the result set or external system. This generally occurs because the name of the field was misspelled, or it was not included in the result set.

&H320E 12814 LCFAIL_INVALID_FIELDNAME Field name
[‘FIELDNAME’] is not valid in this context

The indicated field name is not valid for the context in which it was used. For example, when attempting to create a new metadata object in an external system, this error may be generated if any of the fields are not valid field names for that external system.

&H320F 12815 LCFAIL_VIRTUAL_FIELD Unsupported virtual field
[‘FIELDNAME’]

Virtual fields are an advanced feature only supported by certain Connectors. When using virtual fields, only those fields supported by the Connector in questions are valid. When a Connector supports virtual fields and a field provided has that Connector’s virtual code set, then it will generate this error if it does not understand the supplied virtual field. Check the Connector documentation for information on supported virtual fields.

&H3210 12816 LCFAIL_VIRTUAL_VALUE Invalid data value for virtual field
[‘FIELDNAME’]

Virtual fields are an advanced feature only supported by certain Connectors. When using virtual fields, only values supported by the Connector in questions are valid. When a Connector supports virtual fields and a field provided has that Connector’s virtual code set, then it will generate this error if it cannot handle or interpret the value supplied in that virtual field. Check the Connector documentation for information on supported virtual fields and valid values for those fields..

&H3211 12817 LCFAIL_INVALID_ORDER Invalid order field
[‘FIELDNAME’]

The indicated field was provided as an ordering field, but no such usable field exists in the result set or external system. This error can occur when misspelling a fieldname or leaving the indicated field out of the result set. It can also occur in some cases when using a non-searchable

General Errors

field (as defined by the external systems - check the Connector documentation for information on datatypes that cannot be used as keys) for ordering, and not using an Order metaconnector.

Appendix B

Connector Property Tokens

This appendix provides information about the property tokens used in the Lotus Connectors LotusScript Extensions.

The following flags can be combined with LCX property tokens.

The base value for an LCX property is between 1 and 0xFFFF

Predefined Tokens

LCTOKEN_NAME - Library name for the connector used to create this connection.

LCTOKEN_CONNECTOR_CODE - A code unique to this connector, and the same for all connections to this connector. This code is assigned dynamically, so may change for each execution of a script. This code can be used as a virtual field code to indicate special handling by any connection to this connector.

LCTOKEN_CONNECTION_CODE - A code unique to this connection among all connections. This code is assigned dynamically, so may change for each execution of a script. This code can be used as a virtual field code to indicate special handling by this connection only.

LCTOKEN_EVENT_ERROR - Error code (LCFAIL_XXX constant) to treat as an event. When this error is generated, it is downgraded to an informational event. Set to LCSUCCESS to clear this property.

LCTOKEN_CHARACTER_SET - Character Set indicator representing the character set used by this connector for data transfers to and from the backend system. This is normally dynamically determined by the connector from the backend system at run time. Often read-only, but may be assigned for some connectors. Valid values include any text stream format suffix from the supported character set appendix (e.g., "IBMCP932" for LCSTREAMFMT_IBMCP932.)

LCTOKEN_IGNORE_ERROR - Error code (LCFAIL_XXX constant) to ignore. When this error is generated, it is ignored. Set to LCSUCCESS to clear this property.

LCTOKEN_LCX_VERSION - Version code for the connector LCX library.

General Errors

LCTOKEN_CONNECTOR_NAME - Property representing the first sub-connector for a particular Metaconnector. Setting this property to a connector name is the same as creating a connection of that connector type and assigning it to the metaconnector.

LCTOKEN_SERVER - Server string, used for connectivity.

LCTOKEN_DATABASE - Database string, used for connectivity.

LCTOKEN_USERID - Userid string, used for connectivity.

LCTOKEN_PASSWORD - Password string, used for connectivity. This value is write-only, and cannot be retrieved from a connector once assigned.

LCTOKEN_METADATA - Metadata object name. Used for reading and writing records, as well as creating, dropping, and cataloging metadata and fields.

LCTOKEN_INDEX - Index string. Used when an index name is required, such as when creating or dropping an index.

LCTOKEN_MAP_NAME - Map by name flag - set to TRUE to map fields in fieldlist to backend metadata by name, and to FALSE to map by position.

LCTOKEN_WRITEBACK - Writeback flag - set to TRUE before calling Execute or Select to produce a writeback result set, and before calling Update or Remove to perform a writeback operation against a writeback result set.

LCTOKEN_FIELDNAMES - Selection field name list, a text list (or comma or semicolon separate list of field names in a text string) indicating fields to include in a Select method result set. By default, all fields selected are included; this property can be used to restrict that list.

LCTOKEN_ORDERNAMES - Ordering field name list, a text list (or comma or semicolon separate list of field names in a text string) indicating fields to order a Select method result set by.

LCTOKEN_CONDITION - Condition string in a syntax defined by the connector. This clause will be embedded within a selection or modification statement for Select, keyed Update, or keyed Remove operations.

LCTOKEN_STAMPFIELD - Name of the field containing a timestamp value (must be a datetime datatype). When set before calling the Select method, the result set will be restricted to fields where the timestamp field value is between the value assigned to the BASESTAMP property and returned in the MAXSTAMP property. If BASESTAMP is not set, no minimum will be used. The Select method will determine the current time from the backend server, assign that value to the MAXSTAMP property, and use that datetime as the upper boundary for selection.

LCTOKEN_BASESTAMP - Base timestamp value for timestamp selection - see STAMPFIELD property.

LCTOKEN_MAXSTAMP - Maximum timestamp value for timestamp selection - see the STAMPFIELD property. This property is set by the Connector during Select operations when the STAMPFIELD property is set.

LCTOKEN_TEXT_FORMAT - See CHARACTER_SET property - this property is the numeric constant assigned to the particular character set.

LCTOKEN_PROCEDURE - Procedure name to execute for the Call method.

LCTOKEN_OWNER - Owner name string to restrict Catalog method calls (excluding server and database catalogs) to only objects owned by this owner.

LCTOKEN_SCROLLABLE - Whether to produce a scrollable result set. This property is not supported by any connectors at this time.

Connector Identification Property Tokens

LCTOKEN_IDFLAG_ACTION - Action flags

LCTOKEN_IDFLAG_CONNECTOR - General flags

LCTOKEN_IDFLAG_OBJECT_CATALOG - Catalog flags

LCTOKEN_IDFLAG_OBJECT_CREATE - Create flags

LCTOKEN_IDFLAG_OBJECT_DROP - Drop flags

LCTOKEN_IDNAME_SERVER - Name for server objects in this backend system.

LCTOKEN_IDNAME_DATABASE - Name for database objects in this backend system.

LCTOKEN_IDNAME_USERID - Name for user ID objects in this backend system.

LCTOKEN_IDNAME_PASSWORD - Name for password objects in this backend system.

LCTOKEN_IDNAME_METADATA - Name for metadata objects in this backend system.

LCTOKEN_IDNAME_FIELD - Name for metadata fields in this backend system.

LCTOKEN_IDNAME_ALT_METADATA - Name for alternate metadata objects in this

General Errors

backend system.

LCTOKEN_IDNAME_ALT_FIELD - Name for alternate metadata fields in this backend.

LCTOKEN_IDNAME_PROCEDURE - Name for procedure objects in this backend system.

LCTOKEN_IDNAME_INDEX - Name for index objects in this backend system.

LCTOKEN_IDNAME_PARAMETER - Name for procedure parameters in this backend system.

Appendix C

Connector Properties

This chapter provides information about Connector properties. These properties are used when creating Connections using the Lotus Connectors LotusScript Extensions.

Connector Properties

The table below describes the items included in the tables of Connector properties.

Item	Description
Token	Property token. A name indicates the property is represented by the pre-defined constant LCTOKEN_<name>. For example, METADATA indicates the constant LCTOKEN_METADATA. A number indicates a property specific to the particular Connector.
Name	The property name. This is the dynamic property added through the LSX LC.
Type	<p>The data type of the property:</p> <p><i>Boolean</i> Considered either FALSE (zero) or TRUE (non-zero).</p> <p><i>Int</i> Valid values are indicated in the description.</p> <p><i>Datetime</i> Standard datetime.</p> <p><i>Text</i> Standard text stream.</p> <p><i>Text List</i> Can be submitted as a formatted text list or (more commonly) a comma, semicolon, or newline-separated text value.</p> <p>All default values are FALSE (boolean), 0 (integer), no value (datetime), and the empty string (text) unless otherwise indicated.</p>

Notes Connector Properties

The table below defines the properties for the Lotus Connector for Notes.

Token	Name/Description	Type
SERVER	Server Notes Server. If not supplied, the local Notes database will be accessed.	Text
DATABASE	Database Notes database filepath, not the title. Required. Primarily used for establishing a connection.	Text
METADATA	Metadata Notes form name. Unlike many other Connectors, Notes Execute operations also require a Metadata property value, since the selection formula does not include metadata information.	Text
INDEX	Index Notes view name.	Text
MAP_NAME	MapByName Whether to map by name or position when transferring data between data source and target.	Boolean
WRITEBACK	Writeback Whether to perform writeback or non-writeback operations	Boolean
FIELD_NAMES	FieldNames Text list of fields to select - used for Select operations	Text List
ORDER_NAMES	OrderNames Text list of fields to order results by - used for Select operations	Text List

Token	Name/Description	Type
CONDITION	<p>Condition</p> <p>Notes-specific syntax conditional clause used for Select operations (part of a Notes selection formula). This must be valid syntax for a "select <condition>" formula.</p>	Text
STAMPFIELD	<p>StampField</p> <p>Field name of timestamp field - used for Select operations. Using StampField without a view (DB search instead) only works when the timestamp field is "@Modified", and the LoadModified property is set to TRUE. When not using this new property, a view search is still required (although if no view property is set, a temporary one will be created).</p>	Text
BASESTAMP	<p>BaseStamp</p> <p>Minimum timestamp value to include in results - used for Select operations</p>	Datetime
MAXSTAMP	<p>MaxStamp</p> <p>During a Select operation, set by the Connector to the current/maximum timestamp value - set by Select operations.</p>	Datetime
TEXT_FORMAT	<p>TextFormat</p> <p>This is always LCSTREAMFMT_LMBCS for Notes. Read-only property.</p>	Int
CHARACTER_SET	<p>CharacterSet</p> <p>This is always LCSTREAMFMT_LMBCS for Notes. Read-only property.</p>	Text
1	<p>Port</p> <p>Port name to locate the server on. No value searches all ports.</p>	Text

Notes Connector Properties

Token	Name/Description	Type
2	<p>EnforceForm</p> <p>Whether to enforce the form design on documents created and accessed.</p> <p>0 — The connection does not execute any Notes form- and field-related formulas.</p> <p>1 — Executes all Notes field-related formulas (default value, input translation, and computed field) as part of the session. This option may cause slower data transfer as a result of formula calculations. This option does not execute @Db functions.</p> <p>2 — The session sets all field flags as defined in the form but does not execute any formulas. As a result, special types, indicated in Notes by flags --such as reader and author names--are assigned in new documents as indicated in the form. This option avoids the overhead of computing field formulas.</p>	Int
3	<p>MultiValueAsText</p> <p>Whether to represent all multi-value Notes types (text list, number list, and datetime list) as text values. This can make type transformation between non-Notes Connectors that do not support multi-value types more straightforward.</p>	Boolean
4	<p>BulkStore</p> <p>Whether to buffer all document modifications for bulk submittal at disconnection. This increases performance, but prevents access to those changes until disconnection.</p>	Boolean
5	<p>FilePath</p> <p>Directory on the Domino Server into which attachments are extracted to and attached from when using the FILE virtual field or the LoadFile property. By default, the current working directory is used.</p>	Text
6	<p>UpdateViews</p> <p>Whether to update all views in the database during disconnection. After making extensive changes, the initial opening of a view can perform a time-consuming update.</p>	Boolean
7	<p>CreateDatabase</p>	Boolean

Token	Name/Description	Type
	Whether to create a database during connection if one does not exist. The database is blank, unless a database template is specified with the TemplateServer and TemplateDatabase properties.	
8	TemplateServer Notes Server on which the database specified at TemplateDatabase resides. This template is used in conjunction with the CreateDatabase option. No value uses the local Notes installation	Text
9	TemplateDatabase Notes template database filepath on TemplateServer. This template is used in conjunction with the CreateDatabase option.	Text
10	View Notes view to open for Execute or to use for keyed operations. The view's result set replaces any Statement submitted to Execute.	Text
11	Agent Notes agent to run for an Execute operation on the result set produced by either the Statement parameter or the View property.	Text
12	FullTextQuery Notes full text query for an Execute operation on the result set produced by either the Statement parameter or the View property, and any Agent property.	Text
13	TextMaxLength Notes text data is limited to slightly less than 64K. To simplify data creation on other Connectors with shorter text columns, specify a value to assign as the maximum length for text metadata in any result set produced.	Int
14	FetchViewData Whether to fetch data from view columns rather than from document fields. Requires a View property value.	Boolean
15	AllForms Whether to include documents of all forms. By default, only documents of the form indicated by the metadata	Boolean

Notes Connector Properties

Token	Name/Description	Type
	property value are included. When including documents of other forms, fields in the indicated metadata are used in place of those document's forms.	
16	ViewResponses Whether to include view response documents. By default, only main topics are included. Requires a View property value.	Boolean
17	LoadUnid Whether to add a UNID virtual field to the result set produced by an Execute or Select operation.	Boolean
18	LoadNoteid Whether to add a NOTEID virtual field to the result set produced by an Execute or Select operation.	Boolean
19	LoadRef Whether to add a REF virtual field to the result set produced by an Execute or Select operation.	Boolean
20	LoadFile Whether to add a FILE virtual field to the result set produced by an Execute or Select operation.	Boolean
21	CopyHierarchy Whether to add a LCXHIER virtual field to the result set produced by an Execute operation.	Boolean
22	CopyFile Whether to add a LCXFILE virtual field to the result set produced by an Execute or Select operation.	Boolean
23	ExecuteDelete Whether to delete the result set produced during an Execute operation. This property is to simulate native DELETE statements provided on some other Connectors.	Boolean
24	InsertMail Whether to reroute Inserted documents as mail. Normal Notes addressing information, such as the required SendTo field, is used.	Boolean

Token	Name/Description	Type
25	MailEmbedForm Whether to embed the form in documents mailed through the InsertMail property. Use when the target form is not available in the recipients' mail database.	Boolean
26	Encrypt Encrypts all encryption-enabled fields in the form of the target document. To use private encryption keys (the default is to use the public key), provide one or more private encryption key names, separated by commas or semicolons in EncryptKeyList. These keys must be available on the Domino Server.	Boolean
27	EncryptKeyList One or more private encryption key names, separated by commas or semicolons, to be used by Encrypt. Valid only if Encrypt is True.	Text list
28	PurgeDeleteStubs Clears all deletion stubs from the Notes database when disconnecting. It is strongly recommended that this only be used with UpdateViews, since once the stubs are deleted, they will not be properly removed from views.	Boolean
29	SearchNoCase Makes view searches case-insensitive (they are case-sensitive by default).	Boolean
30	AlterView When the LC LSX does a Select operation with the Connector's View property set, AlterView=True allows the script to modify or create the view if it doesn't already exist or doesn't have the proper formatting. Necessary for select operations when a view and stampfield are specified.	Boolean
31	LoadModified Set to add a field "@Modified" to the result set, which contains the Notes implicit document timestamp. This field can only be read and attempts to set it will be ignored.	Boolean
32	StampViewKey Indicates the value for the first key value for keyed	Int

Notes Connector Properties

Token	Name/Description	Type
	operations when the view is a LC LSX-created view for timestamp selection. Zero means don't use. 1 is the value for docs outside the timestamp selection range, 2 is the value for docs inside the range. When performing a timestamp selection, this is automatically set to 1 by the Connector, and reset to zero when clearing the result set, <i>and should only be used or altered with great care.</i>	

Notes Connector Property Combinations

The following are *invalid* combinations:

- FetchViewData TRUE and Agent or FieldList non-NULL
- View NULL and any of FetchViewData, ViewResponses, or CopyFile TRUE
- ViewResponses TRUE and CopyHierarchy TRUE
- FetchViewData TRUE and any of LoadUnid, LoadNoteid, LoadRef, LoadFile, or CopyFile TRUE
- MapByName TRUE and any of LoadUnid, LoadNoteid, LoadRef, LoadFile, CopyFile, or CopyHierarchy TRUE

When using ordering or timestamps (properties OrderNames or StampField, respectively) in Notes select operations, the View property can be set to indicate the name of the view to use for these operations. When performing selections with View set, the LSX LC will check that the view has the proper formatting (special requirements for either ordered or timestamped result sets). If the view has the correct format, it will be used. Supplying a view name when doing these operations can drastically improve performance by not requiring the database to be reindexed every run. When no view is specified, a temporary view is still created. When the view is the wrong format or doesn't exist, the use of the AlterView property allows the LSX LC to create or overwrite that view to be in its own format.

Notes Virtual Fields

Notes recognizes a set of virtual fields - fields which are interpreted differently by the Notes connector than by other connectors. To use these virtual fields, either set the virtual code of the relevant fields to the Notes connector virtual code (obtain the property `LCTOKEN_CONNECTOR_CODE` from a Notes connection, and set that code into the properly names field using the `LCField.SetVirtualCode` method), or use one of the Notes properties which adds a virtual field to the result set. The virtual fields supported by Notes are described below:

Field Name	Corresponding Property	Data type	Stream Length (in bytes)
UNID	LoadUnid	Binary (BLOB)	16
NOTEID	LoadNoteid	Integer	N/A
REF	LoadRef	Binary (BLOB)	16
FILE	LoadFile	Binary (Text List)	variable (up to 64K)
MODIFIED	LoadModified	Datetime	N/A
LCXHIER	CopyHierarchy	Binary (BLOB)	8
LCXFILE	CopyFile	Binary (BLOB)	4
LCXSPEC	CopyCompSpecial	Binary (BLOB)	4

Descriptions may vary based on whether a document is being read (Fetch) or written (Insert or Update):

UNID

Read - Read the UNID of the document

Write - Set the document's UNID to the field value

NOTEID

Read - Read the NOTEID of the document

Write - ignored

REF

Read - Read the parent UNID of a response document

Write - Make the document a response to the document whose UNID is in this field

FILE

Read - Retrieve all attachments from the document, writing the files to disk. This field is set to a text list containing the filenames of all the files from the document

Write - Store each entry in the text list value as a file attachment to the document.

MODIFIED

Read - Read the last modified timestamp from the document

Write - ignored

LCXHIER

This field is only relevant when the document is both read from and written to a Notes connector. When written, the field contents stored during the read are used to access the source database and transfer the entire response hierarchy beneath the document into the target database. Note that the source result set must still be valid at the time the document is written.

LCXFILE

This field is only relevant when the document is both read from and written to a Notes connector. When written, the field contents stored during the read are used to access the source database and transfer all file attachments directly from the source document to the target document. Since they are transferred directly without temporary storage on disk, this is more efficient than the FILE property for Notes to Notes transfers. The source document must be the most recently fetched document in the result set at the time the document is written.

LCXSPEC

This field is only relevant when the document is both read from and written to a Notes connector. When written, the field contents stored during the read are used to access the source database and transfer specific document fields required to retain full document fidelity for doclinks and rich text fonts. Specifically the fields "\$Links" and "\$Fonts" are copied. The source document must be the most recently fetched document in the result set at the time the document is written.

DB2 Connector Properties

The table below defines the properties for the Lotus Connector for DB2.

Token	Name/Description	Type
DATABASE	Database DB2 Database name. Required. Primarily used for establishing a connection.	Text
USERID	Userid User ID / name: used for logging in during connection	Text
PASSWORD	Password Password: used for logging in during connection	Text
METADATA	Metadata Data table or view: the source or target for data transfer operations.	Text
INDEX	Index Index name.	Text
MAP_NAME	MapByName Whether to map by name (TRUE) or position (FALSE) when transferring data between data source and target.	Boolean
WRITEBACK	Writeback Whether to perform writeback or non-writeback operations	Boolean
FIELD_NAMES	FieldNames Text list of fields to select - used for Select operations	Text List
ORDER_NAMES	OrderNames Text list of fields to order results by - used for Select operations	Text List

Token	Name/Description	Type
CONDITION	Condition DB2-specific syntax conditional clause used for Select, keyed update and remove operations. Must be of valid syntax, as in "select * from table where <condition>".	Text
STAMPFIELD	StampField Field name of timestamp field - used for Select operations	Text
BASESTAMP	BaseStamp Minimum timestamp value to include in results - used for Select operations	Datetime
MAXSTAMP	MaxStamp During a Select operations, set by the Connector to the current timestamp value - set by Select operations	Datetime
TEXT_FORMAT	Text Format Stream format in which text data is transferred between the relevant Connector and its external source. Determined dynamically. Read-only.	Int
CHARACTER_SET	CharacterSet Text format constant suffix for the text format. For example, character set Code Page 932 is represented by the constant LCSTREAMFMT_CP932 value "CP932". Read-only property.	Text
PROCEDURE	Procedure The stored procedure to execute for the Call method.	Text
1	CommitFrequency Number of data modification actions between commits. A value of zero commits at disconnect; one auto-commits after every action; any other value commits after that many data modification actions.	Integer
2	RollbackOnError Whether to rollback the current transaction at disconnection if the session is in an error state.	Boolean
3	CreateMaxLogged Use when creating tables. The maximum length of a	Int

DB2 Connector Properties

Token	Name/Description	Type
	CLOB/BLOB column to create with logging. If the length during metadata creation is longer than this value, the options NOT LOGGED COMPACT are used. A value of zero indicates no maximum, although unbounded columns are always created with this option.	
4	NoJournal Whether the database contains non-journalled data. If so, the transaction isolation level is set to uncommitted read to permit SQL operations.	Boolean
5	CreateInDatabase Use to add "IN DATABASE <dbname>" to a CREATE TABLE query constructed for table creation.	Text

EDA/SQL Connector Properties

The table below defines the properties for the Lotus Connector for EDA/SQL.

Token	Name/Description	Type
SERVER	Server EDA/SQL Server name. Required. Primarily used for establishing a connection.	Text
DATABASE	Database EDA/SQL Engine name. Used to indicate a relational database engine to use.	Text
USERID	Userid User ID / name: used for logging in during connection	Text
PASSWORD	Password Password: used for logging in during connection	Text
METADATA	Metadata Data table: the source or target for data transfer operations.	Text
MAP_NAME	MapByName Whether to map by name or position when transferring data between data source and target.	Boolean
WRITEBACK	Writeback Whether to perform writeback or non-writeback operations	Boolean
FIELD_LIST	FieldList Multi-value text list of fields to select - used for Select operations	Text List
ORDER_LIST	OrderList Multi-value text list of fields to order results by - used for Select operations	Text List
CONDITION	Condition Connector-specific syntax conditional clause - used for	Text

EDA/SQL Connector Properties

Token	Name/Description	Type
	Select operations	
TEXT_FORMAT	<p>Text Format</p> <p>Connector Only: Stream format in which text data is transferred between the relevant Connector and its external source</p>	Int
CHARACTER_SET	<p>CharacterSet</p> <p>Text format constant suffix for the character set to use, if the default of the underlying Connector's character set is to be overridden. For example, for character set Code Page 932, represented by the constant LCSTREAMFMT_CP932, enter "CP932". Read-only property.</p>	Text
1	<p>CommitFrequency</p> <p>Number of data modification actions between commits. A value of zero commits at disconnect; one auto-commits after every action; any other value commits after that many data modification actions.</p>	Integer

File System Connector Properties

The table below defines the properties for the Lotus Connector for File System.

Token	Name/Description	Type
DATABASE	Database Path to the directories (C:\FILES\SOMEFILES or /unix/usr/files). Required.	Text
METADATA	Metadata The subdirectory name (mydirectory or somedir/mydir). This subdirectory contains the files you want to access.	Text
MAP_NAME	MapByName Whether to map by name or position when transferring data between data source and target.	Boolean
WRITEBACK	Writeback Whether to perform writeback or non-writeback operations	Boolean
FIELD_LIST	FieldList Multi-value text list of fields to select - used for Select operations	Text List
ORDER_LIST	OrderList Multi-value text list of fields to order results by - used for Select operations	Text List
CONDITION	Condition The file specification for the files to include. The specification can include the standard wildcard characters: * to match any character string; ? to match any single character.	Text
STAMPFIELD	StampField Field name of timestamp field - used for Select operations	Text
BASESTAMP	BaseStamp Minimum timestamp value to include in results - used for Select operations	Datetime

File System Connector Properties

Token	Name/Description	Type
MAXSTAMP	MaxStamp During a Select operations, set by the Connector to the current timestamp value - set by Select operations	Datetime
TEXT_FORMAT	TextFormat Stream format for file contents.	Int
CHARACTER_SET	CharacterSet Text format constant suffix for the character set to use, if the default of the underlying Connector's character set is to be overridden. For example, character set Code Page 932 is represented by the constant LCSTREAMFMT_CP932 value "CP932". Can be set. When set, it affects the character set of the file contents when they're represented as text (default is NATIVE char set).	Text
1	Binary Whether file contents should be treated as binary (BLOB) data rather than the default (native TEXT).	Boolean
2	Sort Defines the sort order in the case of ordering by filename (0 = binary, 1 = case sensitive, 2 = case insensitive).	Int

File System Metadata

File system metadata is defined as follows:

Filename – Text

Contents – Text

Timestamp – Datetime

Size - int

ODBC Connector Properties

The table below defines the properties for the Lotus Connector for ODBC.

Token	Name/Description	Type
USERID	Userid User ID/name: used for logging in during connection	Text
PASSWORD	Password Password: used for logging in during connection	Text
METADATA	Metadata Data table source or target for data transfer operations	Text
INDEX	Index Data index name.	Text
MAP_NAME	MapByName Whether to map by name or position when transferring data between data source and target.	Boolean
WRITEBACK	Writeback Whether to perform writeback or non-writeback operations	Boolean
FIELD_NAMES	FieldNames Text list of fields to select - used for Select operations	Text List
ORDER_NAMES	OrderNames Text list of fields to order results by - used for Select operations	Text List
CONDITION	Condition Connector-specific syntax conditional clause - used for Select operations	Text
STAMPFIELD	StampField Name of timestamp field used for Select operations	Text
BASESTAMP	BaseStamp	Datetime

Token	Name/Description	Type
	Minimum timestamp value to include in results - used for Select operations	
MAXSTAMP	MaxStamp During a Select operations, set by the Connector to the current timestamp value - set by Select operations	Datetime
TEXT_FORMAT	Text Format Connector Only: Stream format in which text data is transferred between the Connector and its external source	Int
CHARACTER_SET	CharacterSet Text format constant suffix for the character set to use, if the default of the underlying Connector's character set is to be overridden. For example, character set Code Page 932 is represented by the constant LCSTREAMFMT_CP932 value "CP932". Read-only.	Text
1	CommitFrequency Number of data modification actions between commits. A value of zero commits at disconnect; one auto-commits after every action; any other value commits after that many data modification actions.	Integer
2	RollbackOnError Whether to rollback the current transaction at disconnection if the session is in an error state.	Boolean
3	DisableCursor Whether to force disabling of ODBC cursors. This option reduced performance, and should be used only during writeback operations when the ODBC driver inaccurately reports cursor support, resulting in an error.	Boolean
4	QuoteColumn String used to surround column names (for example ")	Text
7	SingleThread By default, the Lotus Connector for ODBC is multi-threaded. However, some ODBC drivers are not multi-threaded. Use this property to force the Connector into a single thread.	Boolean

Oracle Connector Properties

The table below defines the properties for the Lotus Connector for Oracle.

Token	Name/Description	Type
SERVER	Server Oracle Host Name. Required.	Text
USERID	Userid User ID / name - used for logging in during connection	Text
PASSWORD	Password Password - used for logging in during connection	Text
METADATA	Metadata Data table or view: the source or target for data transfer operations.	Text
INDEX	Index Index name.	Text
MAP_NAME	MapByName Whether to map by name or position when transferring data between data source and target.	Boolean
WRITEBACK	Writeback Whether to perform writeback or non-writeback operations	Boolean
FIELD_NAMES	FieldNames Text list of fields to select - used for Select operations	Text List
ORDER_NAMES	OrderNames Text list of fields to order results by - used for Select operations	Text List
CONDITION	Condition Connector-specific syntax conditional clause - used for Select operations	Text

Token	Name/Description	Type
STAMPFIELD	StampField Field name of timestamp field - used for Select operations	Text
BASESTAMP	BaseStamp Minimum timestamp value to include in results - used for Select operations	Datetime
MAXSTAMP	MaxStamp During a Select operations, set by the Connector to the current timestamp value - set by Select operations	Datetime
TEXT_FORMAT	TextFormat Connector only: Stream format in which text data is transferred between the relevant Connector and its external source	Int
CHARACTER_SET	CharacterSet Text format constant suffix for the character set to use, if the default of the underlying Connector's character set is to be overridden. For example, for character set Code Page 932, represented by the constant LCSTREAMFMT_CP932, enter "CP932". Read-only.	Text
PROCEDURE	Procedure The stored procedure to execute for the Call method.	Text
PROCEDURE	Procedure Use when executing parameterized stored procedures in a link-independent fashion. When a session is run and this property is set, the property's value is taken as a stored procedure name and is executed instead of the normal selection operation, potentially producing a result set. If a fieldlist is provided to the Select operation, then all fields in that fieldlist (not just key fields) are supplied to the stored procedure as parameters, with the names taken from the field names.	Text

Oracle Connector Properties

Token	Name/Description	Type
1	<p>CommitFrequency</p> <p>Number of data modification actions between commits. A value of zero commits at disconnect; one auto-commits after every action; any other value commits after that many data modification actions. Note: A commit during a cursored (writeback) operation following an update unlocks result set. Therefore, when a writeback result set is active, the commit frequency property is ignored and commits do not occur.</p>	Integer
2	<p>RollbackOnError</p> <p>Whether to rollback the current transaction at disconnection if the session is in an error state.</p>	Boolean
3	<p>CreateLongColumn</p> <p>Use for table creation. Name of the column to create as a long type during table creation (Oracle tables are restricted to one long column). This value is only used if CreateLongByUser property indicates a user-defined long column.</p>	Text
4	<p>CreateLongByUser</p> <p>Use for table creation. Method of selecting the column to create as a long type during table creation. Oracle tables are restricted to no more than one long column. A value of zero indicates that the Connector should choose the best candidate from the columns being created (the first and/or longest column). A value of one indicates to use the column name in the CreateLongColumn property. A value of two indicates that no long column should be created.</p>	Int

Sybase Connector Properties

The table below defines the properties for the Lotus Connector for Sybase.

Token	Name/Description	Type
SERVER	Server Server name. Required.	Text
DATABASE	Database Database name.	Text
USERID	User ID/name Used for logging in during connection	Text
PASSWORD	Password Password - used for logging in during connection	Text
METADATA	Metadata Metadata object - the source or target for data transfer operations. A form (Notes), table (relational DB), or other data container	Text
INDEX	Index Index name. A view (Notes), index (relational DB), or other data index	Text
MAP_NAME	MapByName Whether to map by name or position when transferring data between data source and target.	Boolean
WRITEBACK	Writeback Whether to perform writeback or non-writeback operations	Boolean
FIELD_LIST	FieldList Multi-value text list of fields to select - used for Select operations	Text List
ORDER_LIST	OrderList Multi-value text list of fields to order results by - used for	Text List

Sybase Connector Properties

Token	Name/Description	Type
	Select operations	
CONDITION	Condition Connector-specific syntax conditional clause - used for Select operations	Text
STAMPFIELD	StampField Field name of timestamp field - used for Select operations	Text
BASESTAMP	BaseStamp Minimum timestamp value to include in results - used for Select operations	Datetime
MAXSTAMP	MaxStamp During a Select operations, set by the Connector to the current timestamp value - set by Select operations	Datetime
TEXT_FORMAT	TextFormat Connector only: Stream format in which text data is transferred between the relevant Connector and its external source	Int
CHARACTER_SET	CharacterSet Text format constant suffix for the character set to use, if the default of the underlying Connector's character set is to be overridden. For example, for character set Code Page 932, represented by the constant LCSTREAMFMT_CP932, enter "CP932". Read-only.	Text
PROCEDURE	Procedure The stored procedure to execute for the Call method. Any returned result set is available.	Text
3	CreateShortBound Whether to truncate bounded text and binary data during table creation. When used, all text and binary fields with a maximum length will be truncated to fit in a char/varchar/binary/varbinary column.	Boolean
4	CreateShortUnbound Whether to truncate unbounded text and binary data during table creation. When used, all text and binary fields with no maximum length will be truncated to fit in a	Boolean

Token	Name/Description	Type
	char/varchar/binary/varbinary column.	
5	DisableCursor Disable Sybase writeback cursors.	Boolean
6	ProcedureStatus A Sybase stored procedure can either generate a result set or return a status code. When a non-zero code is returned, this may signal an error. The default behavior is to interpret the status as a 1-row, 1-column result set. Selecting this option interprets the status code separately, not as a result set (non-zero is error, and zero is success).	Boolean

Appendix D

Character Sets

This appendix provides information about character sets, including character set translation, character sort order, and a list of the character sets supported by the LC LSX.

List of Supported Character Sets

The list is given as text stream format constants. Any of these values may be used as a stream format for a text stream when creating scripts using the Lotus Connectors LotusScript Extensions. To indicate the character set on the local machine, use the constant LCSTREAMFMT_NATIVE.

Stream Format Constant	Description
LCSTREAMFMT_LICS	Lotus International Character Set
LCSTREAMFMT_IBMCP851	MS-DOS PC Greek (CP 851)
LCSTREAMFMT_IBMCP852	MS-DOS PC Eastern European (CP 852)
LCSTREAMFMT_IBMCP853	MS-DOS PC Turkish (CP 853)
LCSTREAMFMT_IBMCP857	MS-DOS PC Turkish (CP 857)
LCSTREAMFMT_IBMCP862	MS-DOS PC Hebrew (CP 862)
LCSTREAMFMT_IBMCP864	MS-DOS PC Arabic (CP 864)
LCSTREAMFMT_IBMCP866	MS-DOS PC Cyrillic Unicode (CP 866)
LCSTREAMFMT_IBMCP437	MS-DOS PC US (CP 437)
LCSTREAMFMT_IBMCP850	MS-DOS PC Western European (CP 850)
LCSTREAMFMT_IBMCP855	MS-DOS PC Cyrillic (CP 855)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_IBMCP860	MS-DOS PC Portuguese (CP 860)
LCSTREAMFMT_IBMCP861	MS-DOS PC Icelandic (CP 861)
LCSTREAMFMT_IBMCP863	MS-DOS PC Canadian French (CP 863)
LCSTREAMFMT_IBMCP865	MS-DOS PC Norwegian (CP 865)
LCSTREAMFMT_IBMCP869	MS-DOS PC Greek (CP 869)
LCSTREAMFMT_IBMCP899	IBM Code Page 899 (CP 899)
LCSTREAMFMT_IBMCP932	MS-DOS PC Japanese Microsoft Shift-JIS (CP 932)
LCSTREAMFMT_IBMCP942	MS-DOS PC Japanese Microsoft Shift-JIS (CP 942)
LCSTREAMFMT_IBMCP891	MS-DOS PC Korean (CP 891)
LCSTREAMFMT_DECMCS	DEC Multinational Character Set
LCSTREAMFMT_EUC	Extended Unix Code
LCSTREAMFMT_KS	MS-DOS Korean - KSC 5601
LCSTREAMFMT_IBMCP949	MS-DOS Korean (CP 949)
LCSTREAMFMT_TCA	TCA
LCSTREAMFMT_BIG5	MS-DOS Taiwan (traditional) Chinese (BIG-5)
LCSTREAMFMT_IBMCP950	MS-DOS Taiwan (traditional) Chinese (CP 950)
LCSTREAMFMT_GB	MS-DOS PRC (simplified) Chinese (GB 2312)
LCSTREAMFMT_IBMCP936	MS-DOS PRC (simplified) Chinese (CP 936)
LCSTREAMFMT_NECESJIS	MS-DOS PC Japanese NEC Shift-JIS (CP 932)
LCSTREAMFMT_ISO646	ASCII
LCSTREAMFMT_ASCII	ASCII
LCSTREAMFMT_ISO88591	ISO Latin-1 US, Western European (ISO-8859-1)
LCSTREAMFMT_IBMCP819	ISO Latin-1 US, Western European (CP 819)

Stream Format Constant	Description
LCSTREAMFMT_ISO88592	ISO Latin-2 Eastern European (ISO-8859-2)
LCSTREAMFMT_IBMCP912	ISO Latin-2 Eastern European (CP 912)
LCSTREAMFMT_ISO88593	ISO Latin-3 Southern European (ISO-8859-3)
LCSTREAMFMT_ISO88594	ISO Latin-4 Northern European (ISO-8859-4)
LCSTREAMFMT_ISO88595	ISO Cyrillic (ISO-8859-5)
LCSTREAMFMT_IBMCP915	ISO Cyrillic (CP 915)
LCSTREAMFMT_ISO88596	ISO Arabic (ISO-8859-6)
LCSTREAMFMT_IBMCP1008	ISO Arabic (CP 1008)
LCSTREAMFMT_ISO88597	ISO Greek (ISO-8859-7)
LCSTREAMFMT_IBMCP813	ISO Greek (CP 813)
LCSTREAMFMT_ISO88598	ISO Hebrew (ISO-8859-8)
LCSTREAMFMT_IBMCP916	ISO Hebrew (CP 916)
LCSTREAMFMT_ISO88599	ISO Latin-5 Southern European (ISO-8859-9)
LCSTREAMFMT_IBMCP920	ISO Latin-5 Southern European (CP 920)
LCSTREAMFMT_HPROMAN	HP Roman (LaserJet)
LCSTREAMFMT_HPGREEK	HP Greek (LaserJet)
LCSTREAMFMT_HPTURKISH	HP Turkish (LaserJet)
LCSTREAMFMT_HPHEBREW	HP Hebrew (Laserjet)
LCSTREAMFMT_HPARABIC	HP Arabic (Laserjet)
LCSTREAMFMT_HPTHAI	HP Thai (Laserjet)
LCSTREAMFMT_HPJAPAN	HP Japanese (Laserjet)
LCSTREAMFMT_HPKANA	HP Kana (Laserjet)
LCSTREAMFMT_HPKOREA	HP Korean (Laserjet)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_HPPRC	HP Simplified Chinese (Laserjet)
LCSTREAMFMT_HPROC	HP Traditional Chinese (Laserjet)
LCSTREAMFMT_IBMCP37	IBM EBCDIC US/Canadian English (CP 37)
LCSTREAMFMT_IBMCP28709	IBM Code Page 28709 (CP28709)
LCSTREAMFMT_IBMCP273	IBM EBCDIC German - Austrian (CP 273)
LCSTREAMFMT_IBMCP278	IBM EBCDIC Finnish, Swedish (CP 278)
LCSTREAMFMT_IBMCP280	IBM EBCDIC Italian (CP 280)
LCSTREAMFMT_IBMCP284	IBM EBCDIC Spanish, Latin American (CP 284)
LCSTREAMFMT_IBMCP285	IBM EBCDIC UK (CP 285)
LCSTREAMFMT_IBMCP290	IBM EBCDIC Japanese (Katakana) (CP 290)
LCSTREAMFMT_IBMCP297	IBM EBCDIC French (CP 297)
LCSTREAMFMT_IBMCP500	IBM EBCDIC International (CP 500)
LCSTREAMFMT_IBMCP277	IBM EBCDIC Danish, Norwegian (CP 277)
LCSTREAMFMT_IBMCP1047	IBM EBCDIC Latin-1 Open Systems (CP 1047)
LCSTREAMFMT_IBMCP1250	Windows Eastern European (CP 1250)
LCSTREAMFMT_IBMCP1251	Windows Cyrillic (CP 1251)
LCSTREAMFMT_IBMCP1252	Windows ANSI (CP 1252)
LCSTREAMFMT_ANSI	ANSI
LCSTREAMFMT_IBMCP1253	Windows Greek (CP 1253)
LCSTREAMFMT_IBMCP1254	Windows Turkish (CP 1254)
LCSTREAMFMT_IBMCP1255	Windows Hebrew (CP 1255)
LCSTREAMFMT_IBMCP1256	Windows Arabic (CP 1256)
LCSTREAMFMT_IBMCP1257	Windows Baltic (CP 1257)

Stream Format Constant	Description
LCSTREAMFMT_IBMCP1363	Windows Korean (CP 1363)
LCSTREAMFMT_MACSCRIPT0	Macintosh Roman (Script 0)
LCSTREAMFMT_MACSCRIPT1	Macintosh Japanese (Script 1)
LCSTREAMFMT_MACSCRIPT2	Macintosh Traditional Chinese (Script 2)
LCSTREAMFMT_MACSCRIPT3	Macintosh Korean (Script 3)
LCSTREAMFMT_MACSCRIPT4	Macintosh Arabic (Script 4)
LCSTREAMFMT_MACSCRIPT5	Macintosh Hebrew (Script 5)
LCSTREAMFMT_MACSCRIPT6	Macintosh Greek (Script 6)
LCSTREAMFMT_MACSCRIPT7	Macintosh Cyrillic (Script 7)
LCSTREAMFMT_MACSCRIPT8	Macintosh Right-left symbol (Script 8)
LCSTREAMFMT_MACSCRIPT9	Macintosh Devanagari (Script 9)
LCSTREAMFMT_MACSCRIPT10	Macintosh Gurmukhi (Script 10)
LCSTREAMFMT_MACSCRIPT11	Macintosh Gujarati (Script 11)
LCSTREAMFMT_MACSCRIPT12	Macintosh Oriya (Script 12)
LCSTREAMFMT_MACSCRIPT13	Macintosh Bengali (Script 13)
LCSTREAMFMT_MACSCRIPT14	Macintosh Tamil (Script 14)
LCSTREAMFMT_MACSCRIPT15	Macintosh Telugu (Script 15)
LCSTREAMFMT_MACSCRIPT16	Macintosh Kannada/Kanarese (Script 16)
LCSTREAMFMT_MACSCRIPT17	Macintosh Malayalam (Script 17)
LCSTREAMFMT_MACSCRIPT18	Macintosh Sinhalese (Script 18)
LCSTREAMFMT_MACSCRIPT19	Macintosh Burmese (Script 19)
LCSTREAMFMT_MACSCRIPT20	Macintosh Khmer/Cambodian (Script 20)
LCSTREAMFMT_MACSCRIPT21	Macintosh Thai (Script 21)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_MACSCRIPT22	Macintosh Laotian (Script 22)
LCSTREAMFMT_MACSCRIPT23	Macintosh Georgian (Script 23)
LCSTREAMFMT_MACSCRIPT24	Macintosh Armenian (Script 24)
LCSTREAMFMT_MACSCRIPT25	Macintosh Simplified Chinese (Script 25)
LCSTREAMFMT_MACSCRIPT26	Macintosh Tibetan (Script 26)
LCSTREAMFMT_MACSCRIPT27	Macintosh Mongolian (Script 27)
LCSTREAMFMT_MACSCRIPT28	Macintosh Geez/Ethiopic (Script 28)
LCSTREAMFMT_MACSCRIPT29	Macintosh EastEurRoman/Slavic (Script 29)
LCSTREAMFMT_MACSCRIPT30	Macintosh Vietnamese (Script 30)
LCSTREAMFMT_MACSCRIPT31	Macintosh extended Arabic/Sindhi (Script 31)
LCSTREAMFMT_MACSCRIPT32	Macintosh uninterpreted symbols (Script 32)
LCSTREAMFMT_MACSCRIPT0CROATIAN	Macintosh Roman variant - Croatian
LCSTREAMFMT_MACSCRIPT0GREEK	Macintosh Roman variant - Greek
LCSTREAMFMT_MACSCRIPT0ICELANDIC	Macintosh Roman variant - Icelandic
LCSTREAMFMT_MACSCRIPT0ROMANIAN	Macintosh Roman variant - Romanian
LCSTREAMFMT_MACSCRIPT0TURKISH	Macintosh Roman variant - Turkish
LCSTREAMFMT_THAI	MS Thai Windows
LCSTREAMFMT_IBMCP874	MS-DOS PC Thai (CP 874)
LCSTREAMFMT_ISO885911	ISO Thai (ISO-8859-11)
LCSTREAMFMT_TIS620	Thai Industrial Standard (TIS620-2529)
LCSTREAMFMT_UNICODE	Unicode (ISO 10646)
LCSTREAMFMT_IBMCP1200	Unicode (IBM CP 1200)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_ISO10646	Unicode (ISO 10646)
LCSTREAMFMT_UTF7	Unicode Transformation Formats 7
LCSTREAMFMT_UTF8	Unicode Transformation Formats 8
LCSTREAMFMT_LMBCS	Lotus MultiByte Character Set (LMBCS)
LCSTREAMFMT_DECNRCUK	DEC National Replacement Char - UK
LCSTREAMFMT_DECNRCDUTCH	DEC Nat'l Replacement Char - Dutch
LCSTREAMFMT_DECNRCFINNISH	DEC Nat'l Replacement Char - Finnish
LCSTREAMFMT_DECNRCFRENCH	DEC Nat'l Replacement Char - French
LCSTREAMFMT_DECNRCFRENCHCAN ADIAN	DEC Nat'l Replacement Char - French Canadian
LCSTREAMFMT_DECNRCGERMAN	DEC Nat'l Replacement Char - German
LCSTREAMFMT_DECNRCITALIAN	DEC Nat'l Replacement Char - Italian
LCSTREAMFMT_DECNRCNORWEGIAN DANISH	DEC Nat'l Replacement Char - Norwegian Danish
LCSTREAMFMT_DECNRCPORTUGUES E	DEC Nat'l Replacement Char - Portuguese
LCSTREAMFMT_DECNRCSPANISH	DEC Nat'l Replacement Char - Spanish
LCSTREAMFMT_DECNRCSWEDISH	DEC Nat'l Replacement Char - Swedish
LCSTREAMFMT_DECNRCSWISS	DEC Nat'l Replacement Char - Swiss
LCSTREAMFMT_T61	Teletex T.61
LCSTREAMFMT_T50	Teletex T.50
LCSTREAMFMT_ASN1	ANSI Standard Notation (ASN.1)
LCSTREAMFMT_IBMCP856	MS-DOS PC Hebrew (CP 85)
LCSTREAMFMT_IBMCP1004	MS-DOS PC Desktop Publishing (CP 1004)
LCSTREAMFMT_IBMCP1002	IBM EBCDIC DCF (CP 1002)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_IBMCP1003	IBM EBCDIC US Text Subset (CP 1003)
LCSTREAMFMT_IBMCP1025	IBM EBCDIC Cyrillic (CP 1025)
LCSTREAMFMT_IBMCP1026	IBM EBCDIC Turkish (CP 1026)
LCSTREAMFMT_IBMCP1028	IBM EBCDIC Hebrew Publishing (CP 1028)
LCSTREAMFMT_IBMCP256	IBM EBCDIC International #1 (CP 256)
LCSTREAMFMT_IBMCP259	IBM EBCDIC Symbols Set 7 (CP 259)
LCSTREAMFMT_IBMCP274	IBM EBCDIC Belgian (CP 274)
LCSTREAMFMT_IBMCP275	IBM EBCDIC Brazilian (CP 275)
LCSTREAMFMT_IBMCP281	IBM EBCDIC Japanese (Latin) (CP 281)
LCSTREAMFMT_IBMCP282	IBM EBCDIC Portugese (CP 282)
LCSTREAMFMT_IBMCP361	IBM EBCDIC International #5 (CP 361)
LCSTREAMFMT_IBMCP382	IBM EBCDIC Austrian, German, Switzerland (CP 382)
LCSTREAMFMT_IBMCP383	IBM EBCDIC Belgian (CP 383)
LCSTREAMFMT_IBMCP384	IBM EBCDIC Brazilian (CP 384)
LCSTREAMFMT_IBMCP385	IBM EBCDIC Canadaian (French) (CP 385)
LCSTREAMFMT_IBMCP386	IBM EBCDIC Danish, Norwegian (CP 386)
LCSTREAMFMT_IBMCP387	IBM EBCDIC Finnish, Swedish (CP 387)
LCSTREAMFMT_IBMCP388	IBM EBCDIC French, Swiss (CP 388)
LCSTREAMFMT_IBMCP389	IBM EBCDIC Italian, Swiss (CP 389)
LCSTREAMFMT_IBMCP390	IBM EBCDIC Japanese (Latin) (CP 390)
LCSTREAMFMT_IBMCP391	IBM EBCDIC Portugese (CP 391)
LCSTREAMFMT_IBMCP392	IBM EBCDIC Spanish, Phillipines (CP 392)
LCSTREAMFMT_IBMCP393	IBM EBCDIC Latin American (Spanish Speaking) (CP 393)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_IBMCP394	IBM EBCDIC UK, Australian, Hong Kong, Ireland, New Zealand (CP 394)
LCSTREAMFMT_IBMCP395	IBM EBCDIC US, Canadian (English) (CP 395)
LCSTREAMFMT_IBMCP423	IBM EBCDIC Greek 183 (CP 423)
LCSTREAMFMT_IBMCP424	IBM EBCDIC Hebrew (CP 424)
LCSTREAMFMT_IBMCP803	IBM EBCDIC Hebrew Character Set A (CP 803)
LCSTREAMFMT_IBMCP870	IBM EBCDIC Eastern Europe (CP 870)
LCSTREAMFMT_IBMCP871	IBM EBCDIC Icelandic (CP 871)
LCSTREAMFMT_IBMCP875	IBM EBCDIC Greek (CP 875)
LCSTREAMFMT_IBMCP880	IBM EBCDIC Cyrillic (CP 880)
LCSTREAMFMT_IBMCP905	IBM EBCDIC Turkish (CP 905)
LCSTREAMFMT_IBMCP948	IBM Extended Taiwanese (CP 948)
LCSTREAMFMT_IBMCP938	IBM Taiwanese (CP 938)
LCSTREAMFMT_IBMCP1381	IBM GBK = GB + Hanzi (CP 1381)
LCSTREAMFMT_IBMCP1386	IBM Traditional Chinese (CP 1386)
LCSTREAMFMT_EACC	East Asian Character Code Set (ANSI Z39.64-1989)
LCSTREAMFMT_JIS	Japanese Information Standard 0201 (JIS 201)
LCSTREAMFMT_CCCCII	Chinese Character Code for Information Interchange (Taiwan)
LCSTREAMFMT_XEROXCJK	Xerox CJK
LCSTREAMFMT_IBMCP944	IBM Extended Korean (CP 944)
LCSTREAMFMT_IBMCP934	IBM Korean (CP 934)
LCSTREAMFMT_IBMCP737	MS-DOS PC Greek (CP 737)
LCSTREAMFMT_IBMCP775	MS-DOS PC Baltic (CP 775)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_ISO6937	Latin chars (non-spacing accents) similar to T.61
LCSTREAMFMT_BASE64	Content-Transfer-Encoding
LCSTREAMFMT_JIS2	Japanese Information Standard 0208 (JIS 208)
LCSTREAMFMT_EUCJ	Extended Unix Code - Japanese
LCSTREAMFMT_EUCT	Extended Unix Code - Taiwanese
LCSTREAMFMT_EUCK	Extended Unix Code - Korean
LCSTREAMFMT_ISOKR	ISO-2022-KR switching: treated as EUCK
LCSTREAMFMT_EUCC	Extended Unix Code - Chinese
LCSTREAMFMT_IBMCP921	Replacement for Lithuanian (CP 921)
LCSTREAMFMT_IBMCP922	Russian (CP 922)
LCSTREAMFMT_KOI8	Cyrillic Internet Support
LCSTREAMFMT_IBMCP720	IBM Code Page 720 (CP 720)
LCSTREAMFMT_IBMCP1258	Windows Vietnamese (CP 1258)
LCSTREAMFMT_ISO885910	ISO Latin-6 (ISO-8859-10)
LCSTREAMFMT_JP1TEXT	OSI/JIS X 5003-1987 X.400 Japanese ISP
LCSTREAMFMT_VIQRI	Vietnamese Quoted Readable
LCSTREAMFMT_VISCII	Vietnamese VISCII 1.1 (VICSII)
LCSTREAMFMT_VISCII1	TCVN Vietnamese Orthographic (VCSII-1)
LCSTREAMFMT_VISCII2	TCVN Vietnamese Graphic (VCSII-2)* /
LCSTREAMFMT_IBMCP838	IBM EBCDIC SBCS Thai (CP 838)
LCSTREAMFMT_IBMCP9030	IBM EBCDIC SBCS Thai (CP 9030)
LCSTREAMFMT_IBMCP833	IBM EBCDIC SBCS Korean - extended (CP 833)
LCSTREAMFMT_IBMCP836	IBM EBCDIC SBCS PRC (simplified) Chinese (CP 836)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_IBMCP1027	IBM EBCDIC SBCS Japanese Latin - extended (CP 1027)
LCSTREAMFMT_IBMCP420	IBM EBCDIC Arabic (CP 420)
LCSTREAMFMT_IBMCP918	IBM EBCDIC Code Page 918 (CP 918)
LCSTREAMFMT_IBMCP1097	IBM EBCDIC Code Page 1097 (CP 1097)
LCSTREAMFMT_IBMCP1112	IBM EBCDIC Code Page 1112 (CP 1112)
LCSTREAMFMT_IBMCP1122	IBM EBCDIC Code Page 1122 (CP 1122)
LCSTREAMFMT_IBMCP1123	IBM EBCDIC Code Page 1123 (CP 1123)
LCSTREAMFMT_IBMCP1129	IBM EBCDIC Code Page 1129 (CP 1129)
LCSTREAMFMT_IBMCP1130	IBM EBCDIC Code Page 1130 (CP 1130)
LCSTREAMFMT_IBMCP1132	IBM EBCDIC Code Page 1132 (CP 1132)
LCSTREAMFMT_IBMCP1133	IBM EBCDIC Code Page 1133 (CP 1133)
LCSTREAMFMT_IBMCP930	IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 930)
LCSTREAMFMT_IBMCP933	IBM EBCDIC EUC Korean Mixed (CP 933)
LCSTREAMFMT_IBMCP935	IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 935)
LCSTREAMFMT_IBMCP937	IBM EBCDIC EUC Taiwan (traditional) Chinese Mixed (CP 937)
LCSTREAMFMT_IBMCP939	IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 939)
LCSTREAMFMT_IBMCP931	IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 931)
LCSTREAMFMT_IBMCP1388	IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 1388)
LCSTREAMFMT_IBMCP5026	IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 5026)
LCSTREAMFMT_IBMCP5035	IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 5035)
LCSTREAMFMT_IBMCP300	IBM EBCDIC DBCS Japanese (CP 300)

List of Supported Character Sets

Stream Format Constant	Description
LCSTREAMFMT_IBMCP834	IBM EBCDIC DBCS Korean (CP 834)
LCSTREAMFMT_IBMCP835	IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 835)
LCSTREAMFMT_IBMCP837	IBM EBCDIC DBCS PRC (simplified) Chinese (CP 837)
LCSTREAMFMT_IBMCP930X	IBM EBCDIC DBCS Japanese (CP 930X)
LCSTREAMFMT_IBMCP933X	IBM EBCDIC DBCS Korean (CP 933X)
LCSTREAMFMT_IBMCP935X	IBM EBCDIC DBCS PRC (simplified) Chinese (CP 935X)
LCSTREAMFMT_IBMCP937X	IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 937X)
LCSTREAMFMT_IBMCP939X	IBM EBCDIC DBCS Japanese (CP 939X)
LCSTREAMFMT_IBMCP931X	IBM EBCDIC DBCS PRC (simplified) Chinese (CP 931X)
LCSTREAMFMT_IBMCP1388X	IBM EBCDIC DBCS PRC (CP 1388X)
LCSTREAMFMT_IBMCP1383	IBM Traditional Chinese (CP 1383)
LCSTREAMFMT_IBMCP806	ISO Devnagiri (CP 806)
LCSTREAMFMT_IBMCP1137	IBM EBCDIC Devnagiri (CP 1137)
LCSTREAMFMT_VISCII3	TCVN3 Vietnamese (VCSII-3)
LCSTREAMFMT_TCVN3	TCVN3 Vietnamese (VCSII-3)